Universidad Carlos III de Madrid
Escuela Politécnica Superior
Grado en Ingeniería Informática
Mención en Computación
Planning and Learning Group

Integración de los Robots Humanoides REEM y REEM-C en la Arquitectura Robótica RoboComp

Trabajo de Fin de Grado

por

Andrea Haro Casas

21 de Junio de 2016

Tutor:
José Carlos González Dorado

Cotutor:
Fernando Fernández Rebollo

TRABAJO DE FIN DE GRADO

INTEGRACIÓN DE LOS ROBOTS HUMANOIDES REEM Y REEM-C EN LA ARQUITECTURA ROBÓTICA ROBOCOMP

Autor: ANDREA HARO CASAS
Tutor: JOSÉ CARLOS GONZÁLEZ DORADO
Cotutor: FERNANDO FERNÁNDEZ REBOLLO

TRIBUNAL

Presidente: José María Valls Ferrán
Secretario: Isidro Hernanz Cabilla
Vocal: José Carlos Castillo Montoya

Tras el acto de defensa y lectura el día 7 de Julio de 2016 en la **Escuela Politécnica Superior** de la **Universidad Carlos III de Madrid** (Leganés), el tribunal le otorga la siguiente **CALIFICACIÓN**:

Agradecimientos

Después de casi 6 años desde que comencé la aventura universitaria, por fin doy por finalizada esta bonita etapa con la entrega de este trabajo. Por supuesto he de agradecer a todas esas personas que han hecho que esta aventura haya sido inolvidable.

En primer lugar a Fernando por presentarme este proyecto y engancharme con él desde el primer momento. A mi tutor José Carlos, por la ayuda, dedicación y adaptación a mis difíciles horarios en estos últimos meses sacrificando los viernes para ayudarnos a avanzar con el trabajo. En general a todo el grupo de PLG de la Universidad Carlos III por el apoyo, energía y estupendo ambiente de trabajo que trasmiten. A la empresa PAL Robotics en general, y a Sam, Victor y Francesco en particular por la ayuda facilitada y la rapidez en contestar a nuestras consultas que han facilitado la resolución de más de una complicaciones.

En el ámbito personal, a mi familia por todo el apoyo brindado y la paciencia conmigo durante todos estos años en los que había días que no había quien me entendiera por el estrés y los agobios.

A mis amigos, en especial Vero y Elea que me han dado tantas fuerzas, ánimo y conseguían despejar mi cabeza.

A Julio, mi compañero de vida por los buenos momentos que pasamos juntos, por mimarme, escucharme, apoyarme y animarme siempre.

Al grupito de compañeros de aventura que desde luego sin ellos no habría sido lo mismo. Georgi, Maroto, Rubén, Jorge, Victor, Rober,... por los piques, las risas, la ayuda siempre y los buenos momentos de estos años incluso trabajando. Espero todo lo mejor para vosotros.

Por último, a mi monito Carlos, uno de los grandes descubrimientos de la aventura. Casi compañeros por descarte en un primer momento pero después inseparables (agradecer a Victor la recomendación de juntarnos). Muchas gracias por aguantarme, por llevarme a la calma en los momentos de drama o de modo furby, por las risas, los consejos, por ayudarme en todo momento y porque no había mejor broche final para esta etapa que hacer este último trabajo juntos. Sabes que aquí tienes una amiga para siempre.

¡Gracias a todos!

Por favor, cita este trabajo como:

Andrea Haro Casas: *Integración de los Robots Humanoides REEM y REEM-C en la Arquitectura RoboComp, Trabajo de Fin de Grado, Universidad Carlos III de Madrid, 2016.*

Abstract

Traditional methods for rehabilitation therapies require continuous attention from therapists during rehabilitation sessions and are quite monotonous for patients, especially young patients, the children. To facilitate the therapist work and motivate patients it has been developed the NAOTherapist project in the field of robotics. In this project, based on NAOTherapist, twho new components are added.

This document presents the bachelor thesis *Integration of humanoids robots REEM and REEM-C in robotic architecture RoboComp* for the University Carlos III de Madrid, in the Polytechnic School of Leganés.

The developed work creates two new components that are integrated into the architecture NAOTherapist replacing the original NAO robot by the robot REEM and REEM-C robots. The new component maintains the compatibility with the rest of the architecture and adapts all existing functionality. To maintain this functionality it has been developed an adaptive function of the movement of new robots, and the creation of an interface that provides the ability to interact with the patient and to solve the morphological and functional differences between NAO and REEM robots.

This document describes the process followed in the development of the new components, as well as the performed evaluation using the original domain of NAOTherapist architecture, rehabilitation therapies with children.

Keywords

Robotics, Retargeting, Rehabilitation Therapies, Robotic Frameworks, REEM, ROS, RoboComp.

Resumen

Los métodos tradicionales para realizar terapias de rehabilitación requieren una atención continua por parte de los terapeutas durante la realización de las sesiones y son bastante monótonas para los pacientes, sobre todo los pacientes de corta edad, los niños. Para facilitar la labor de terapeuta y motivar a los pacientes se ha desarrollado, en el ámbito de la robótica el proyecto NAOTherapist, base del presente documento y al que se añaden dos nuevos componentes.

Este documento muestra el Trabajo de Fin de Grado realizado "Integración de los robots humanoides REEM y REEM-C en la arquitectura robótica RoboComp" en la Universidad Carlos III de Madrid, en la Escuela Politécnica Superior de Leganés.

El trabajo desarrollado crea dos nuevos componentes que se integran en la arquitectura NAOTherapist sustituyendo al robot NAO original por el robot REEM o el robot REEM-C. Estos dos nuevos componentes son totalmente compatibles con el resto de la arquitectura, adaptándose a la funcionalidad ya existente en la misma. Para realizar esta adaptación se ha desarrollado una función de adaptación del movimiento de los nuevos robots, se crea una interfaz que proporcione la capacidad de interacción con el paciente y resuelva correctamente las diferencias entre el robot inicial y los nuevos a nivel morfológico y funcional.

En este documento se detalla todo el proceso seguido en el desarrollo de los dos nuevos componentes y se realiza la evaluación de los mismos utilizando el dominio original de la arquitectura NAOTherapist, las terapias de rehabilitación con niños.

Palabras clave

Robótica, *Retargeting*, Terapias de Rehabilitación, *Frameworks* robóticos, REEM, ROS, RoboComp.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se introduce el contexto que enmarca el desarrollo del presente proyecto.

Los métodos tradicionales para realizar terapias de rehabilitación requieren una atención continua por parte de los terapeutas durante la realización de sesiones de terapia y son bastante monótonas para los pacientes, sobre todo los pacientes de corta edad, los niños. Para facilitar la labor de terapeuta y motivar a los pacientes se ha desarrollado, en el ámbito de la robótica el proyecto NAOTherapist, base del presente documento. La labor de este trabajo es llevar a cabo la integración de dos nuevo componentes **ReemComp** y **ReemCompC** en la arquitectura NAOTherapist [González et al. 2015].

Debido a la amplitud del desarrollo del presente proyecto, sobre todo en algunas de sus tareas, el trabajo de desarrollo del componente ReemComp se ha realizado conjuntamente entre Carlos Manzano Carrasco [Manzano Carrasco 2016] y la autora de este proyecto. Dadas algunas de las primeras fases de desarrollo, inevitablemente comunes y la gran cantidad de dependencias entre ambos proyectos, existe al menos un grado del 20 % de implicación en algunos componentes del desarrollo. Los proyectos están en-

focados hacia distintos dominios, tratando el presente proyecto sobre THERAPIST[1], dominio inicialmente propuesto. Tras este desarrollo común, en el presente proyecto se ha llevado a cabo la adaptación de un nuevo componente adicional ReemCompC, que se desarrolla con la siguiente versión del robot REEM. Además se ha desarrollado una mejora de la interfaz cambiando el diseño y proporcionando la posibilidad del cambio de los modelos visibles en ella. Se ha llevado a cabo la implementación de un control de secuencia de la ejecución del sistema y la organización de los componentes para que con un único fichero se pueda ejecutar uno y otro componente según se desee.

A continuación, se realiza una descripción de la motivación para la realización de este proyecto. Seguidamente se exponen, de forma concisa, los objetivos concretos de este trabajo.

En la última sección de este capítulo, se realizará una descripción de la estructura seguida en este documento, junto con una distribución porcentual de la carga de trabajo de cada uno de los integrantes del equipo del proyecto (Tabla 1.1).

1.1. Contexto

Se definen las terapias como una parte de la medicina que se ocupa del tratamiento de las enfermedades. En el desarrollo de las sesiones de terapia se trata de prevenir y manejar desórdenes que impliquen complicaciones en el movimiento humano. En este ámbito en este trabajo nos centramos en dos enfermedades que acusan en mayor medida a los niños, la parálisis braquial obstétrica y la parálisis cerebral infactil que afectan a las extremidades superiores de los pacientes.

La **Parálisis Braquial Obstétrica** (PBO) es una debilidad o pérdida de movimiento de las extremidades superiores producida cuando el conjunto de nervios alrededor del hombro (denominados plexo braquial) se daña durante el nacimiento de un ser humano (ver Figura 1.1).

[1] http://www.therapist.uma.es/

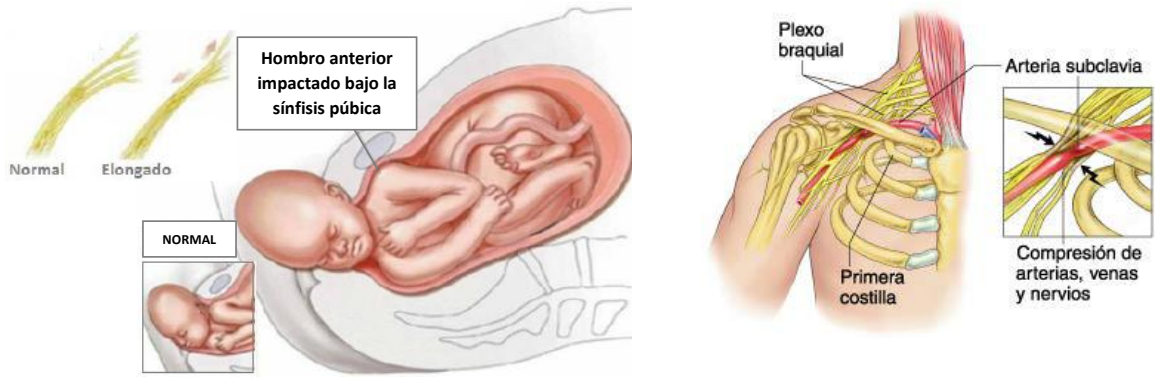


Figura 1.1: Elongación durante el parto del plexo braquial (izquierda) y plexo braquial (derecha).

Gracias a los avances de la medicina y la mejora de las técnicas durante el parto, esta enfermedad se ha reducido drásticamente. Aun así, se estima que 1,5 de cada 1000 nacimientos presentan este problema. Además, no es exclusivo del parto, ya que otras causas, como los accidentes de tráfico, pueden producir esta lesión. Una parte importante de la cura de esta enfermedad es la rehabilitación física para recuperar total o parcialmente la movilidad del brazo afectado [Limthongthang et al. 2013]

La **Parálisis Cerebral Infantil** (PCI) es una enfermedad cerebral no progresiva que impide disponer del completo control de las funciones motoras del cuerpo [Krägeloh-Mann et al. 2009]. Como ocurre con la Parálisis Braquial Obstétrica, esta afección se produce principalmente por complicaciones durante el parto, aunque también se puede producir a causa de un accidente. Un 60 % de los niños con deficiencias motoras están diagnosticados con esta enfermedad. A pesar de no ser un defecto común, existen suficientes casos como para no llegar a ser considerada una enfermedad rara, teniendo una tasa de un 0,2 % de casos cuyos síntomas coinciden con esta enfermedad [Castelli 2011].

Estas dos enfermedades están relacionadas directamente con las extremidades superiores, produciendo una limitación de autonomía del paciente en su vida diaria. En ambos casos, no existe cura conocida para las enfermedades, por lo que un paciente afectado deberá convivir con ello toda la vida. Sin embargo, la rehabilitación juega un papel muy importante en este caso, ya que si bien no se puede llegar a curar completamente, si se puede mejorar enormemente la calidad de vida de la persona. Gracias a una buena rehabilitación el paciente es capaz de fortalecer la musculatura y acostumbrar al cerebro sobre el modo de mover las extremidades afectadas [Hensey 2009].

THERAPIST es un proyecto del Plan Nacional de Investigación (TIN2012-38079) en el que trabajan varios grupos de investigación de varias universidades españolas en colaboración con un grupo experto de terapeutas del Hospital Universitario Virgen del Rocío de Sevilla. El objetivo de este proyecto es desarrollar terapias de rehabilitación mediante el uso de robótica social de asistencia, en la que se utilizará un robot para realizar parte de la labor del terapeuta. Gracias a la colaboración del Hospital Universitario Virgen del Rocío se tiene acceso tanto al protocolo terapéutico interno del hospital como a pacientes reales. Los pacientes que participan en este proyecto son menores de edad de entre 3 y 14 años que padecen estas dos enfermedades descritas.

De este proyecto inicial, nace **NAOTherapist**, que es la base del presente proyecto. El proyecto NAOTherapist ha desarrollado un sistema de terapias de las extremidades superiores utilizando el robot NAO[2] como el que se puede observar en la Figura 1.2. Gracias a la Planificación Automática, que se define como una técnica de inteligencia artificial que sirve para encontrar una secuencia de acciones (plan) que cumpla las precondiciones establecidas, partiendo de un estado inicial, con un dominio con un conjunto de acciones posible y un estado meta, se desarrollan terapias de rehabilitación automáticas personalizadas para cada uno de los paciente. Primeramente se planifican los ejercicios a realizar según el paciente y el robot le indica las posturas

[2] http://www.aldebaran.com/en/humanoid-robot/nao-robot/ (Accedido por última vez 25/02/2016)

que debe ir realizando corrigiéndole en caso de ser necesario. En el Capítulo 2 de este documento se expondrá en mayor detalle este proyecto, describiendo cada uno de los componentes de la arquitectura.

Figura 1.2: Robot NAO.

Destacar, que la arquitectura de NAOTherapist es lo bastante general como para que su uso pueda también estar orientado a otros dominios, como por ejemplo para realizar el juego del Simón [Turp et al. 2015]. Éste es un juego de memoria que originalmente consiste en recordar una serie de colores y repetirla, si se acierta dicha serie aumente la secuencia de colores a repetir. Indicar también que esta arquitectura es modulable por lo que en el presente proyecto se pretende conseguir una integración de la arquitectura NAOTherapist con los robots REEM y REEM-C incluyendo dos nuevos componentes. Estos dos robots deben ser capaces de realizar las terapias e interactuar con el paciente dotando a la terapia de un valor añadido distinto y seguramente atractivo para la mayoría de los pacientes a los que va dirigido el proyecto.

1.2. Motivación

Se ha comprobado que las terapias para este tipo de pacientes suelen ser largas y bastante aburridas, consistiendo en repetidos movimientos de las articulaciones superiores para fortalecer las extremidades afectadas. Esta monotonía en los ejercicios suele producir una pérdida de interés, sobre todo en los pacientes de menor edad, los niños. Éstos requieren una mayor motivación y atención por parte del personal terapéutico para conseguir que no se distraigan y así poder aprovechar en mayor medida el tiempo asignado a la rehabilitación del paciente [Calderita et al. 2013]. También se ha comprobado, que en algunos casos, a pesar de los esfuerzos no se logran los objetivos de la terapia, por lo que supone un punto negativo en el ámbito económico y profesional, debido al coste que supone la dedicación del personal especializado [Meyer-Heim et al. 2013]. La perdida de motivación y los escasos resultados pueden afectar a la vida del paciente y con ello la de sus familiares.

Según la motivación original del proyecto NAOTherapist (base del presente proyecto), los pacientes con mayor pérdida de interés eran los niños y por ello se tomó la decisión de utilizar el robot NAO, con una altura de 58 cm y atractivo a la vista de un niño, de tal modo que pueda parecerle una entidad social con la cual pueda relacionarse y con ello convertir la terapia en un entretenimiento. Se ha observado que debido al tamaño del robot NAO, en algunas posiciones se puede dar lugar a que el paciente se equivoque, como por ejemplo en una postura en el que el robot NAO señala al frente, el paciente tiende a señalar al robot (mas abajo por se el niño más alto que él.) pensando que el robot le estaba señalando a él. Vistos estos casos, surge la idea de utilizar un robot mayor, más grande, que sea más semejante al ser humano como son los robots REEM y REEM-C que pueden verse en la Figura 1.3. De acuerdo a la motivación inicial del proyecto NAOTherapist, las enfermedades tratadas afectan a extremidades superiores, por ello el robot REEM a pesar de no disponer de piernas, puede llegar a ser un fantástico robot terapeuta ya que su estabilidad es uno de sus puntos fuertes, su base hace que sea casi imposible la perdida de equilibrio y caída del

robot. Por otro lado el robot REEM-C es lo mas semejante a un ser humano pudiendo reproducir igualmente la terapia original de las extremidades superiores, pero además facilitando la incorporación futura de rehabilitación en extremidades inferiores igual que en el NAO.

Figura 1.3: Robot REEM y REEM-C.

Indicar que el uso de robots grandes para la realización de terapias es también muy útil para la rehabilitación de adultos, no solo con niños, ya que el robot podría mostrarle la postura deseada claramente, en lugar de mostrarlas en una pantalla, aprovechando se este modo la capacidad de visión en profundidad de las personas. Es ventajoso también por que no todas las poses indicadas en algunas sesiones son tan evidentes en una pantalla 2D.

Las ventajas de realizar las terapias con robots son muchas, por ejemplo, como ya se ha indicado, se reduce el coste de personal especializado durante las sesiones. A pesar de no ser un sistema 100 % autónomo, ya que su principal objetivo no es sustituir al especialista, sino facilitar el trabajo de este y complementarlo. Por otro lado, como ya se ha comentado, las enfermedades que se han tenido en cuenta para el desarrollo no tienen cura definitiva a día de hoy.

Por todo lo comentado, el objetivo del presente proyecto es desarrollar dos nuevos componente dentro de la arquitectura NAOTherapist, de tal forma que sea el robot REEM y REEM-C (independientemente, no a la vez) los que se puedan utilizar en el sistema y por ello sean éstos los nuevos terapeutas. Se quiere que los componentes tengan compatibilidad completa con la arquitectura original, por lo que se ha implementado el movimiento de cada uno de los robots y el sonido de éstos, así como un apoyo visual para facilitar la interacción del usuario con estos robots y aprovechar también algunas de sus capacidades.

Para conseguir este objetivo, es imprescindible el desarrollo de un sistema de *retargeting* para estos robots, que permita adaptar la postura del paciente al robot y viceversa. Es decir que el robot se capaz de realizar la misma postura que visualiza del paciente, por ejemplo colocando en un ángulo de 90º el brazo derecho respecto al torso. En la arquitectura NAOTherapist original, las poses estaban almacenadas como puntos del modelo de un esqueleto humano capturados a través de un sensor 3D y adaptados para su uso con el robot NAO. En este trabajo se crea una función basada en la del robot NAO que permita a los nuevos robots tener las mismas capacidades del robot NAO.

Además, se hace indispensable incluir un sistema que reproduzca funcionalidades de la arquitectura con el robot NAO, tales como los botones que proporcionan el control sobre la terapia (inicio, pausa,...) y los ojos con leds de colores que aportan un feedback al paciente indicando si la postura que realiza es correcta. Por este motivo, en el robot REEM se ha desarrollado una interfaz gráfica que permita simular esta funcionalidad. En el robot REEM-C, dado que no dispone de una pantalla, se podrá conectar un dispositivo que muestre dicha interfaz. Adicionalmente se ha desarrollado un sistema de reproducción de ficheros de audio y reproducción de audio a partir de un texto, lo que proporciona ayuda al usuario y facilita la interacción humano robot.

1.3. Objetivos del trabajo

En esta sección se definen los objetivos concretos de este trabajo:

1. Comprender y analizar el funcionamiento de la arquitectura del proyecto de investigación NAOTherapist. Integrar el framework robótica ROS (*Robot Operating System*) y RoboComp (framework utilizado en el proyecto NAOTherapist), indispensable para que los nuevos robots REEM y REEM-C pueda comunicarse con la arquitectura original.
2. Conocer e implementar la ejecución de movimientos simples del robot REEM y REEM-C en el simulador Gazebo a través de comandos simples en Python.
3. Investigar el dominio y posibilidad de movimiento de cada una de las articulaciones del robot REEM y REEM-C e implementar una función de transformación de los ángulos del esqueleto de un ser humano extraído a partir de los datos de un sensor 3D a los ángulos de las articulaciones del robot REEM y REEM-C (*retargeting*). Con ello se conseguirá que los robots puedan seguir los mismos movimientos que un humano frente a un sensor 3D.
   a) Selección de articulaciones para cada robot sobre las que realizar el *retargeting*.
   b) Transformación de los ángulos de forma que paciente (esqueleto 3D) y robot puedan realizar los mismos movimientos
4. Implementación y mejora de una interfaz gráfica que pueda simular la funcionalidad del robot original NAO.
5. Integrar la arquitectura y movimiento del robot REEM en la arquitectura NAOTherapist.
6. Integrar la arquitectura y movimiento del robot REEM-C en base al componente adquirido con el robot REEM.

1.4. Estructura del documento

La memoria del trabajo está estructurada de tal manera que se va describiendo de los conceptos más generales a los conceptos más concretos.

En el presente capítulo se muestra una visión general del proyecto desarrollando una introducción al mismo, junto con la motivación y los objetivos finales del desarrollo del trabajo.

El Capítulo 2 desarrolla el estado de la cuestión, es decir, se expone el estado actual de los principales elementos del sistema. Se muestra una visión panorámica de trabajos científicos que tienen relación con diferentes aspectos del proyecto que se desarrolla, así como una explicación de ciertas técnicas y herramientas necesarias para el desarrollo del documento.

En el Capítulo 3 se documenta el análisis y diseño del sistema. En primer lugar se describen las normas y restricciones establecidas para la implementación del sistema. A continuación se detalla el entorno operacional sobre el cual se realiza el proyecto. Seguidamente, se especifican los requisitos y casos de uso del sistema y finalmente, se indica la metodología utilizada para la realización del proyecto.

En el Capítulo 4, se detalla la implementación del sistema realizada, indicando los principales componentes desarrollados. Seguidamente, en el Capítulo 5, se realiza la evaluación del sistema, detallando las pruebas realizadas para comprobar el comportamiento del sistema y los resultados obtenidos. En el Capítulo 6 se realiza la planificación y presupuestos estimados para la realización de este proyecto.

Por último, se incluyen en el Capítulo 7 las conclusiones tras la realización del proyecto y las posibles líneas futuras que se podrían seguir para continuar con el desarrollo.

Al final de este documento se añaden varios anexos con un resumen en inglés y el manual de instalación de todos los componentes del sistema.

Como ya se ha mencionado en la introducción, el trabajo está complementado por el proyecto realizado por Carlos Manzano Carrasco. A pesar de estar desarrollado en dominios distintos, presenta numerosas dependencias y puntos de confluencia, de modo que para algunas tareas y la buena finalización del proyecto, ha sido esencial la cooperación entre ambos. La carga de trabajo se ha repartido de forma equilibrada distribuida tal y como se muestra en la Tabla 1.1.

TAREA	Andrea Haro	Carlos Manzano
Análisis del sistema	**50%**	50%
Diseño del sistema	**50%**	50%
Implementación del sistema		
Integración de ROS en NAOTherapist	**50%**	50%
Comprobación conexión ROS	**100%**	0%
Manejo de excepciones y scripting	**100%**	0%
Interfaces adicionales	**100%**	0%
Comandos simples en Python	**50%**	50%
Implementación retargeting	**60%**	40%
Integración REEM-C	**100%**	0%
Desarrollo interfaz	**30%**	70%
Comunicación de ReemComp con la interfaz	**20%**	80%
Reproducción de audio	**0%**	100%
Text-to-Speech	**0%**	100%
Dominios de aplicación y evaluación		
Pruebas y ejecución NAOTherapist	**100%**	0%
Pruebas y ejecución juego Simon	**0%**	100%

Tabla 1.1: Reparto de carga de trabajo

Capítulo 2

Estado de la cuestión

A continuación se describen las principales tecnologías y herramientas usadas durante el desarrollo del trabajo, con el objetivo de facilitar la comprensión de la implementación que se detallará en el Cápitulo 4. Es decir, se enmarca el presente trabajo en el estado estado actual en el que se encuentra.

Para comenzar, se hace una pequeña introducción de robots terapeutas, a continuación se hace una detalla el origen del proyecto THERAPIST, Ursus y NAOTherapist que son la base del presente proyecto. Seguidamente se explica el funcionamiento y utilidad del framework robótica RoboComp utilizado en la arquitectura NAOTherapist.

Seguidamente, se detallan y explican las tecnologías y herramientas utilizadas: Kinect, REEM, REEM-C y framework robótico ROS junto con el simulador Gazebo. Por último se describe el concepto de *retargeting*, parte indispensable de este trabajo.

2.1. Robots terapeutas

El término *robot* procede de la palabra checa *robota*, que significa "trabajo forzado" o "servidumbre" y su primera aparición data del año 1921 en la obra *Rossum's Universal Robots* del autor Karet Capek [Capek 1921]. La obra fue un éxito y en seguida se estrenó un multitud de teatros de Europa y estados Unidos. El termino *robótica* define a la ciencia relacionada con inteligencia artificial y con la ingeniería mecánica, es decir define las acciones de razonar y ejecutar movimientos físicas sugeridas por el razonamiento previo. Este término surge en 1942 por el escritor, bioquímico y divulgador científico Isaac Asimov en su novela corta *Runaround* donde el autor describe las tres leyes de la robótica.

Desde entonces, empezó a crecer el ámbito de la robótica y así sigue siendo en la actualidad desarrollando multitud de tipos de robots para todo tipo de entornos y situaciones, desde pequeños juguetes, brazos robóticos para usar en cadenas de montaje en fabricas e incluso robots para enviar a explorar planetas, como son los *rover* que envía la NASA al espacio para investigación.

Entre todos los ámbitos en los que se utiliza la robótica hoy en día, uno de los que se pueden destacar es el ámbito terapéutico y de rehabilitación de personas, que es el contexto que enmarca el presente proyecto. Un terapeuta es aquella persona que dispone de unas habilidades especiales obtenidas a través de la formación y la experiencia, en una o más áreas dentro de la asistencia sanitaria, siendo su misión ofrecer apoyo a personas que así lo requieran por su situación física o mental. El terapeuta es el encargado de llevar a cabo distintas sesiones, que constituyen una terapia, con la finalidad de mejorar la calidad de vida de la persona a la que ayuda, el paciente. Por tanto, en este entorno, se define un robot terapeuta como aquel que gracias a la experiencia adquirida, bien sea por aprendizaje automático o por la ayuda de un terapeuta humano, es capaz de desarrollar la terapia diseñada por el experto terapeuta y así ayudar a este con su labor. Además, este nuevo concepto de robot terapeuta facilita al paciente las sesiones, haciéndolas más amenas, entretenidas y dándoles una pincelada

de diversidad respecto a las terapias convencionales. Indicar que el robot terapeuta no sustituye al terapeuta humano sino que le ayuda con su tarea mejorando por tanto las terapias que desarrolla.

En la actualidad existen bastantes ejemplos del uso de los robots terapeutas para multitud de áreas de la medicina. Por ejemplo, se está diseñando un robot terapeuta que facilite la movilidad y las terapias de rehabilitación en los miembros inferiores [Humberto et al. 2013]. Para las terapias de rehabilitación de miembros inferiores se está diseñando un robot que ayude al movimiento que se desea conseguir con la pierna para que esta vaya cogiendo movilidad. En la Figura 2.1 el diseño del robot con el que se pretende sustituir parte de la tarea del terapeuta ya que el movimiento del robot podrá definirse de forma más exacta de acuerdo a las necesidades del paciente. En este caso, aún es un estudio que está en proceso pero que se pretende llegar a implementar por el bajo coste que supondría y las grandes mejoras en las terapias de miembros inferiores.

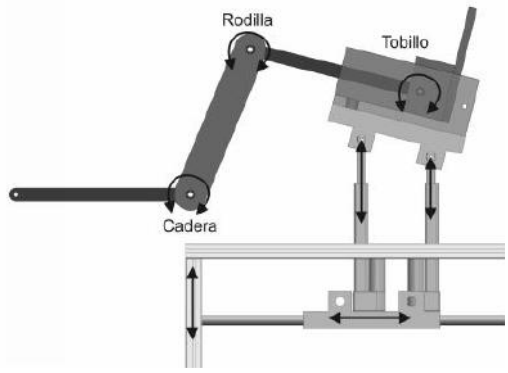


Figura 2.1: Diseño robot terapias miembros inferiores

Otra muestra de robots terapeutas es la del robot AMADEO[1] (ver Figura 2.2) diseñado y utilizado expresamente para las terapias de las manos y los dedos. El robot simula el movimiento de agarra natural del ser humano y ejecuta secuencias de

[1] http://tyromotion.com/en/products/amadeo Accedido última vez 16/05/2016

movimientos automatizados. Dependiendo del nivel de daño que tenga el paciente, se puede tratar de forma pasiva o activa pudiendo hacer multitud de ejercicios e incluso de forma interactiva con juegos. Sobre este robot se ha hecho un estudio clínico en el departamento de neurocirugía de El Instituto de Hospitalización y atención de la Ciencia (IRCCS) San Raffaele Pisana (Roma) [Sale et al. 2012] con 12 pacientes que se prestaron voluntarios a una terapia de 6 semanas con este robot. Los resultados muestran una mejora en múltiples medidas de rendimiento de la articulación, y todos los pacientes toleraron bien la terapia, sin complicaciones.

Figura 2.2: Robot AMADEO

Por otro lado, científicos japoneses apuestan por las terapias con animales robóticos, por lo que se le da al robot una forma más amigable para realizar la terapia. Un ejemplo de robot terapeuta animal es el robot Paro[2] (ver Figura 2.3), conocido en España como robot Nuca, que puede ayudar a personas ancianas con alzheimer o niños que padecen autismo o se encuentran hospitalizados [Burton 2013]. Por su forma

[2]urlhttp://www.parorobots.com/index.asp Accedido última vez 16/05/2016

y aspecto amigable este robot es muy beneficioso para este tipo de pacientes. El robot tiene capacidad de aprendizaje, por lo que les sirve para reducir el estrés y estimular la interacción y socialización del paciente.

Figura 2.3: Robot Paro

Siguiendo la evolución de la forma del robot, se encuentran estudios con robots medio humanoides que se centran en estudiar el posible papel de un robot como herramienta para el seguimiento de actividades cognitivas de personas mayores o con demencia por ejemplo con un juego de música. En el juego, se le pide al usuario a encontrar y pulse el botón que corresponde a la canción que se está reproduciendo, y, en algunos casos a decir el nombre de la canción, y cantar. El robot se encarga de describir el juego al paciente, animar e interactuar con él [Tapus et al. 2009].

Orientado a terapias con niños, el autor Fridin utiliza el robot NAO en terapias con niños autistas [Fridin et al. 2011]. En este trabajo se idea y propone una arquitectura bastante sofisticada aunque finalmente sin implementar. Diseña un prototipo en el cual el robot mueve los brazos frente a una fila de niños para un entrenamiento con ellos de forma secuencial. El objetivo es analizar como el niño acepta la orden del robot y crea compenetración con él.

Continuando con esta línea relacionada con la rehabilitación de los niños, se encuentra el proyecto THERAPIST del que surge NAOTherapist, base del presente proyecto y de los que se profundiza en la siguiente sección.

2.2. Proyecto THERAPIST

Como ya se ha mencionado en el primer capítulo de este documento, el proyecto THERAPIST surge con el objetivo de crear un nuevo sistema de rehabilitación para ayudar a pacientes con edades comprendidas entre 3 y 14 años que padecen Parálisis Cerebral Infantil y Parálisis Braquial Obstétrica. Estas dos enfermedades, son incurables, pero gracias al entrenamiento y rehabilitación de las extremidades se puede conseguir mejorar notablemente la calidad de vida de los pacientes. Recalcar que el objetivo no es sustituir por completo al terapeuta, si no utilizar al robot como una herramienta que sirva de apoyo a la terapia, ya que necesita supervisión aunque ahorre bastante tiempo.

En sus inicios, el proyecto se denominaba Ursus pero no se llegó a desarrollar completamente y dio lugar al proyecto THERAPIST que realiza las terapias con una versión del robot Ursus mejorada [Suárez Mejías et al. 2013], dotando con este nombre a la idea originaria del proyecto. En esta primera fase experimental con el robot Ursus, se demostró que los pacientes disfrutaban con las innovadoras sesiones de rehabilitación. Pero el robot carecía de la autonomía suficiente, ya que este estaba programado según una máquina de estados y por tanto no era capaz de planificar las terapias ni conocer nuevas formas de relación con el paciente. Este sistema no facilitaba la labor del especialista terapeuta y se determinó desarrollar una amplificación de la idea original que dio lugar al proyecto THERAPIST[3]. En la la parte izquierda de la Figura 2.4 se puede visualizar el robot Ursus, que como se observa es un robot humanoide con formada de oso. En esa misma figura a la derecha, se muestra la evolución de dicho robot para el proyecto THERAPIST. El proyecto THERAPIST tiene como objetivo principal la labor de dotar de mayor autonomía y capacidad social al robot usado en Ursus, incluyendo técnicas de inteligencia artificial como el Aprendizaje Automático y la Planificación Automática. Al aumentar la autonomía e interacción social era necesario investigar e incluir nuevas técnicas que sirvieran de base y permitieran una correcta

[3]http://www.therapist.uma.es/ Accedido por última vez 01/02/2016

Figura 2.4: Robot Ursus (izquierda) y evolución para proyecto THERAPIST (derecha).

aplicación del nuevo sistema. De este modo, se podría extraer y utilizar más información de los sensores de la cámara Kinect para que la planificación del comportamiento fuera más coherente.

Tanto en THERAPIST como en Ursus, el protocolo médico seguido tiene dos fases. En la primera de ellas, una pantalla muestra un vídeo de una persona reproduciendo un ejercicio. Seguidamente, el robot reproduce ese mismo ejercicio con sus extremidades y solicita al paciente que repita sus movimientos. En el caso de que el paciente se equivoque, el robot es dirigido para que le enseñe al paciente a realizar el ejercicio correctamente. En la segunda fase, el paciente juega con algunos juegos prediseñados de realidad aumentada que le obligan a reproducir los movimientos terapéuticos pero esta vez de forma mas amena.

Para hacer estos ejercicios, se hace uso uso de la cámara 3D Kinect de Microsoft, que es un controlador de juego libre y entretenimiento diseñado por Microsoft para la videoconsola Xbox 360. El dispositivo permite a los usuarios controlar e interactuar con la consola sin la necesidad del contacto físico, reconoce gestos, posturas y comandos de voz. El sistema utiliza el *framework* robótico RoboComp[4], que es un sistema basado en componentes reutilizables.

[4]http://robocomp.sourceforge.net/ (Accedida el 04/04/2016)

En base a THERAPIST, surge el proyecto NAOTherapist con un nuevo robot, el robot NAO. En la siguiente sección se detalla más acerca de este último, que es además la base del presente proyecto.

2.2.1. Proyecto NAOTherapist

Del proyecto THERAPIST, surge el proyecto NAOTherapist, en el cual se incluye al robot NAO como terapeuta. NAOTherapist es una arquitectura robótica que contiene un sistema de diseño y supervisión de terapias de rehabilitación.

El sistema esta basado en Planificación Automática que es una disciplina de inteligencia artificial que tiene como objetivo la generación de planes, es decir una planificación, generalmente para su ejecución por un robot o agente automático. Los sistemas de planificación que contienen estos algoritmos se denominan planificadores y constan de 3 parámetros: un estado inicial del mundo, una descripción del objetivo a alcanzar (meta) y conjunto de acciones posibles.

El sistema en primer lugar planifica una serie de posturas especificadas según las necesidades o patología concretas de cada paciente. A continuación, un robot humanoide, el robot NAO, guía al paciente enseñándole las poses que debe tomar y corrigiéndole en caso de que no realice correctamente la postura indicada.

En el presente proyecto se crean dos nuevos componentes que sustituyen el componente del robot inicial NAO (componente *NAORobot*), por lo que alguno de los nuevos robots podrá ser el que realice la serie de ejercicios planificados y llevar la interacción necesaria para terminar la sesión. En la Figura 2.5 se muestra la arquitectura inicial al completo y a continuación se detallan los principales componentes:

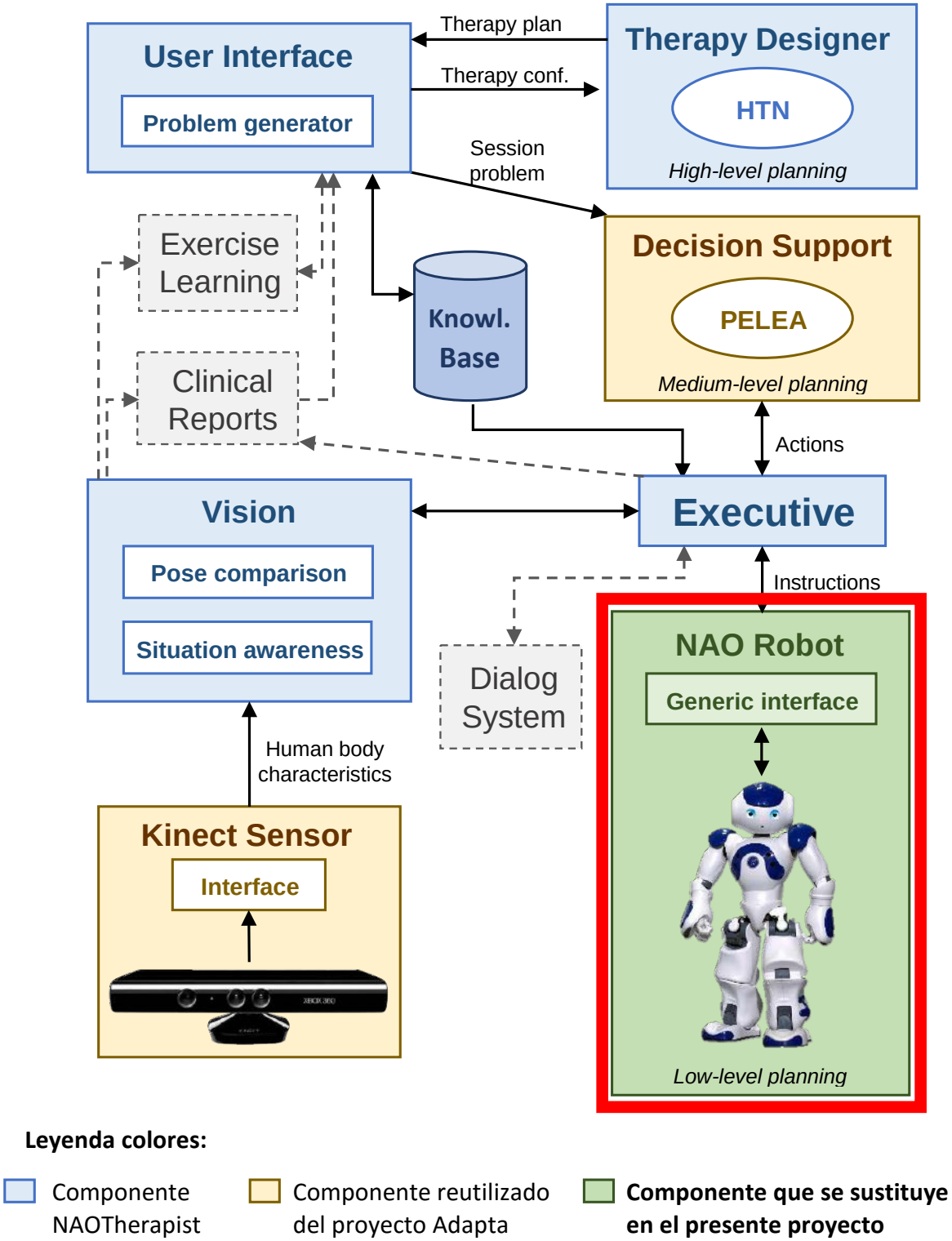


Figura 2.5: Arquitectura NAOTherapist original

- **Diseñador de terapias (*User Interface y Therapy Desgigner*)**: este componente muestra una interfaz gráfica que permite configurar de la terapia que se va a realizar en función del diagnostico del paciente. El diseñador se plantea como una herramienta de apoyo al terapeuta para facilitarle la labor de planificación de la terapia asegurando el cumplimiento de los objetivos clínicos. Se define un plan de terapias como un conjunto de sesiones, que a su vez se componen de diferentes ejercicios o posturas que el usuario debe imitar. Los ejercicios se dividen en varias fases: calentamiento, entrenamiento y enfriamiento, siendo los ejercicios de la fase de entrenamiento los mas intensos. Por tanto, antes de la rehabilitación, el terapeuta configura el plan de la terapia, definiendo el numero de sesiones y ejercicios que debe seguir el paciente y se envía dicha configuración al diseñador de terapias como un problema de Planificación Automática [Ghallab et al. 2004].

  El planificador automático encuentra las acciones necesarias para transitar desde el estado inicial a un estado final que cumple la meta establecida respetando todas las precondiciones de las acciones. Comentar que esta parte de la arquitectura no va ha ser utilizada para este trabajo.

- **PELEA (*Decision Support*)**: PELEA (*Planning, Execution and Learning Architecture*) [Alcázar et al. 2010] es una subarquitectura ideada para la integración de planificación automática, ejecución y monitorización, aunque no es exclusiva del campo de la robótica. Tiene un diseño modular, de forma que es posible personalizarla para añadir o quitar módulos según se desee. Está integrada dentro del *framework* de RoboComp para utilizarla en la arquitectura de NAOTherapist con 3 de sus módulos (ver Figura 2.6).

  La unión entre PELEA y el resto de la arquitectura la lleva a cabo el módulo Execution (no confundir con el componente Ejecutivo de NAOTherapist (este Execution es módulo de PELEA). PELEA es el componente encargado de generar un plan de acciones e ir devolviéndolas una a una, a petición. Cada acción tiene una serie de requisitos que deben cumplirse en el estado del mundo. Si al

componente le llega un estado que no es el esperado, replanifica y genera otro plan de acciones.

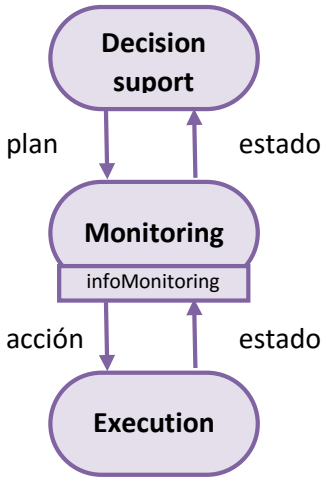


Figura 2.6: Diagrama de funcionamiento básico de PELEA.

Según el diagrama de la Figura 2.6 el funcionamiento básico es el siguiente: El módulo Execution recibe una petición del componente Ejecutivo con el estado actual del mundo y se lo envía al módulo Monitoring. En el caso de que el estado recibido y esperado sean iguales, Monitoring busca en la variable infoMonitoring cual es la siguiente acción que se debe ejecutar, que ha sido previamente planificada y se la manda a Execution. En el caso de que el estado actual sea diferente del esperado, el plan inicial ya no es válido, por lo que Monitoring solicita un nuevo plan a Decision Support. Este último, llama a un planificador automático externo que interpreta el plan recibido y lo regenera. Una vez regenerado el plan, se lo envía a Monitoring que lo almacena en su variable infoMonitoring. Finalmente Monitoring manda la siguiente acción a Execution. PELEA está conectado al componente Ejecutivo de la arquitectura NAOTherapist, encargado de pedir y realizar las acciones.

- **Ejecutivo (*Executive*)**: en esencia, el componente Ejecutivo es el encargado del control de una sesión. Para ello, se establece comunicación con PELEA y con el componente de Vision. El Ejecutivo solicita una acción a PELEA, que se encarga de planificar y retornar una acción. Cuando se obtiene esta nueva acción, se mandan al robot las instrucciones necesarias para ejecutarla. Para que se pueda solicitar una acción, Ejecutivo debe tener una visión clara del estado actual del mundo, es por ello que se utiliza el componente de Vision. Dependiendo de la información recogida por los sensores se crea un estado del mundo con una serie de predicados en formato PDDL (Planning Domain Definition Language) [McDermott et al. 1998] que se mandan a PELEA para que realice la planificación de la siguiente acción.

- **Vision (*Vision*)**: Obtiene información sobre la postura y estado del paciente. Con ayuda de los sensores de la Kinect se obtienen datos característicos del paciente, como son el esqueleto del cuerpo y la posición de las manos. Vision está compuesto principalmente por dos elementos. El primero de ellos **Pose comparison**: tomando como referencia el esqueleto obtenido con la Kinect, calcula la diferencia entre los valores de los puntos característicos del esqueleto y la postura que tiene el paciente. Con esta diferencia se determina el grado de diferencia y se corrige al paciente en caso de ser necesario. El segundo componente, **Situation awareness** detecta si el paciente está situado en el área de entrenamiento, si se encuentra de pie o sentado e incluso distraído. Gracias a esto, el robot puede llamar la atención del paciente para proseguir con la terapia en caso de que no esté preparado.

- **Interfaz robot (*NAORobot*)**: NAO Robot es el componente del robot en sí, y es quien recibe del Ejecutivo las acciones que se deben ejecutar, enviando al robot las instrucciones necesarias para ejecutar dicha acción. En el presente trabajo, esta interfaz ha sido completamente sustituida para que la arquitectura se puede usar con el robot REEM o con el robot REEM-C, en vez de con el robot NAO, sin ser necesario cambiar con ningún otro componente de la arquitectura.

Clinical Reports [Martín et al. 2015], *Dialog System y Exercise Learning* son componentes de la arquitectura que aún se encuentran en desarrollo y no son utilizados por NAOTherapist, por lo que no se hace referencia a ellos en este documento.

Para el presente proyecto se utilizaran todos los componentes que contiene la arquitectura NAOTherapist, exceptuando el diseñador de terapias que no será necesario, el robot que se ha sustituido y la nueva interfaz desarrollada para mantener la misma funcionalidad que el robot original.

2.2.2. Framework robótico RoboComp

RoboComp [Manso et al. 2010] es el *framework* robótico que se usa en Ursus y posteriormente en THERAPIST. También se utiliza este *framework* en NAOTherapist.

Este *framework* está formado por varios componentes independientes ideado para que pueda ser distribuido en varios equipos. La comunicación entre los distintos componentes es establecida a través del middleware Ice (Internet Communications Engine)[5]. Estos componentes, tan solo deben implementar una interfaz Ice que interactúa con el middleware y es éste quien les facilita la comunicación mediantes eventos TCP/IP. Este tipo de arquitectura del *framework* es muy útil dado que existen componentes que actúan mejor en sistemas operativos de Linux (como son por ejemplo planificadores y controladores hardware) y otros tienen una mejor funcionalidad en sistemas operativos Windows (como puede ser el caso de la cámara Kinect de Microsoft). De hecho,

[5] http://www.zeroc.com/overview.html Accedido por última vez 05/04/2016

el proyecto de NAOTherapist funciona a la misma vez en dos equipos diferentes con sistemas operativos distintos: en uno de ellas se utiliza Windows que es el dedicado al componte RoboComp de la Kinect y otro con el sistema operativo Ubuntu para los demás componentes. Cabe destacar, que con este *framework* pueden utilizarse componente programados en diferentes lenguajes de programación, siendo utilizados por ejemplo para el proyecto NAOTherapist los lenguajes de Java, Python y C.

El *framework* RoboComp puede solucionar problemas de escalabilidad, reusabilidad y flexibilidad que suelen aparecer en el ámbito de robótica con la programación orientada a componentes. Es decir, gracias a este *framework* se pueden crear componentes totalmente reutilizables en distintas arquitecturas, no únicamente en la arquitectura para la que fue ideada, además de permitir la integración de componentes externos como es el caso de este proyecto.

Los robots utilizados para este proyecto, REEM y REEM-C funcionan bajo el *framework* ROS, a diferencia del proyecto NAOTherapist en el que se utiliza RoboComp, por lo tanto se ha tenido que llevar a cabo una integración de los nuevos componentes en la arquitectura NAOTherapist. En el Capítulo 6 del presente documento se puede examinar la complejidad del estudio, comprensión y determinación de los pasos a seguir debido al tiempo necesitado para su finalización, como se puede ver en la Tabla 6.1.

2.3. Tecnologías utilizadas

En esta sección se describen las principales herramientas y tecnologías empleadas en el desarrollo del proyecto. Servirán de guía para la comprensión del Capítulo 4 en el que se describe la implementación del sistema que esta esta fuertemente consolidada gracias a estas tecnologías y algoritmos estudiados.

2.3.1. Kinect

El sensor 3D Kinect desarrollado por Microsoft comienza a comercializarse a finales de 2010 con el objetivo de poder controlar la consola Xbox 360 mediante el reconocimiento de voz y gestos. Más tarde, se añadió la compatibilidad del sensor con sistemas Windows y poco después se consiguió un controlador de código libre para su uso en GNU/Linux. Posteriormente, se desarrolló una nueva versión (Kinect 2) con la finalidad de utilizarla en la consola Xbox One.

En el proyecto NAOTherapist se está utilizando actualmente la primera versión de la Kinect (versión de 2010) por lo que para este proyecto se usará esa misma versión. Todas las referencias sobre "Kinect" serán sobre dicha versión.

La Kinect, tal y como se puede ver en la Figura 2.7 esta compuesta por una cámara con sensor RGB, un sensor infrarrojo, un sensor de profundidad, micrófono y un acelerómetro. A continuación se define la funcionalidad de cada uno de estos componentes:

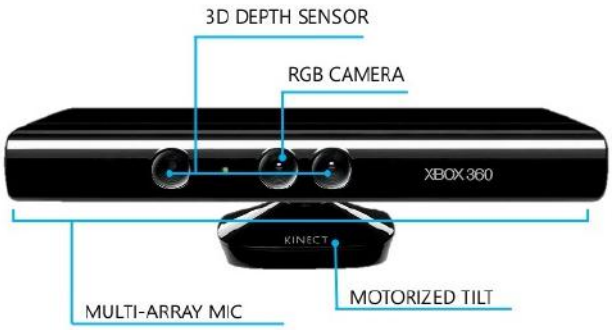


Figura 2.7: Elementos del sensor 3D Kinect.

- **Sensor infrarrojo y sensor de profundidad:** Gracias a estos sensores se consigue medir la profundidad y distancia del objeto, permitiendo dar la forma correcta a la imagen. Este sensor es imprescindible en NAOTherapist para poder reconocer correctamente las posturas de las personas. En 2D no sería suficiente.

- **Cámara RGB:** Es una cámara tradicional que captura imágenes a color (gracias al sensor RGB) con una resolución de 1280x960 píxeles a 30 imágenes por segundo.
- **Micrófono múltiple:** El controlador cuenta con 4 micrófonos con los cuales se graba el audio y la dirección de donde proviene el sonido.
- **Acelerómetro:** Gracias a este componente se consigue la orientación de la Kinect sin importar si está en una superficie no plana.

Gracias al bajo coste del controlador, su sencilla utilización y su buen funcionamiento, han hecho que se convierta en el sensor 3D más utilizado en el ámbito universitario y de investigación.

El uso de este controlador fuera del ámbito de los juegos, suele estar asociado al reconocimiento de objetos, gestos faciales o el esqueleto humano para su aplicación por ejemplo en planificación de caminos o interacción humano-robot.

La Kinect se ha utilizado en robots escaladores, gracias a algoritmos de planificación automática y el uso de este sensor para conocer el estado actual del robot y los siguientes puntos de ascenso [Dung et al. 2014]. Otro ejemplo de uso puede ser el creador de mapas en 2D desarrollado por el Instituto Tecnológico Sepuluh Nopember de Indonesia [Mardiyanto et al. 2015]. Gracias al uso de este controlador, un robot puede de crear el mapa de un recinto mientras se encuentra en movimiento, almacenando la información de las zonas recorridas anteriormente.

Por supuesto, este componentes es fundamental para el desarrollo del proyecto NAOTherapist y del presente trabajo, utilizado principalmente para el reconocimiento de posturas.

2.3.2. Robot REEM

El robot REEM (ver Figura 2.8) es un robot humanoide que nace en 2012 en la empresa española Pal Robotics[6]. Esta empresa, con sede principal en Barcelona, esta formada por un equipo de ingenieros que diseñan, ensamblan y personalizan robots humanoides de servicio que son capaces de navegar de forma autónoma en entornos reales.

Figura 2.8: Robot REEM.

El robot tiene el tamaño de una persona, concretamente mide 1,70 m y pesa 100 kg. La parte inferior del robot es una base móvil con ruedas que le permiten desplazarse fácilmente sin riesgo de caídas por pérdida de equilibrio. Esta base sobresale por la parte trasera para transportar objetos en dicha superficie. La parte superior del robot, es mas semejante a una persona. Dispone de cabeza, torso, brazos, muñecas y manos que puede mover gracias a distintos motores. Cuenta con 32 grados de libertad en la parte superior (7 en cada brazo, 7 en cada mano, 2 en el torso y 2 en la cabeza) y

[6]http://pal-robotics.com/es/

FIGURA	ARTICULACIÓN	MÁXIMO (º)	MÍNIMO (º)
(1)	GiroHombro	118º	-13º
(2)	AperturaHombro	178º	-44º
(3)	GiroCodo	156º	-134º
(4)	AperturaCodo	123º	0º
(5)	GiroAntebrazo	11º	-118º
(6)	GiroMuñeca	88º	-88º
(7)	AperturaMuñeca	88º	-88º
(8)	Dedos	Pulgar: 88º Resto: 248º	Pulgar: 0º Resto: 0º
(9)	GiroTorso	73º	-73º
(10)	InclinaciónTorso	34º	-13º
(11)	GiroCabeza	73º	-73º
(12)	InclinaciónCabeza	34º	-13º

Tabla 2.1: Valores mínimo y máximo de las articulaciones de REEM.

otros dos en la parte inferior situados en la base móvil. En el presente trabajo, se usan sobre todo las articulaciones relacionadas con los brazos, aunque en las animaciones también se ejecutan movimientos con todas las articulaciones superiores. Se puede ver la definición de cada una de las articulaciones en la Figura 2.9 junto con los valores máximo y mínimo de giro descritos en la Tabla 2.1

El robot cuenta con una desviación de la articulación del hombro respecto al torso, lo que hace que al girar completamente no se muestre del mismo modo la pose que en el robot NAO. En la Figura 2.10 se puede ver este efecto, en verde el ángulo sobre el que se mueve el robot NAO y en color rojo la desviación producida por el diseño del robot REEM. Esta diferencia de morfología hace que con una apertura de 90º la postura del brazo en el REEM no quede perpendicular al torso, situación que si se produce en el NAO, por lo tanto se han desarrollado varias transformaciones para corregir esta diferencia entre los esqueletos de ambos robots. En el Capítulo 4 se explican en mayor detalle todas las transformaciones desarrolladas.

Adicionalmente, el robot tiene dos ordenadores integrados. Uno de ellos se encarga

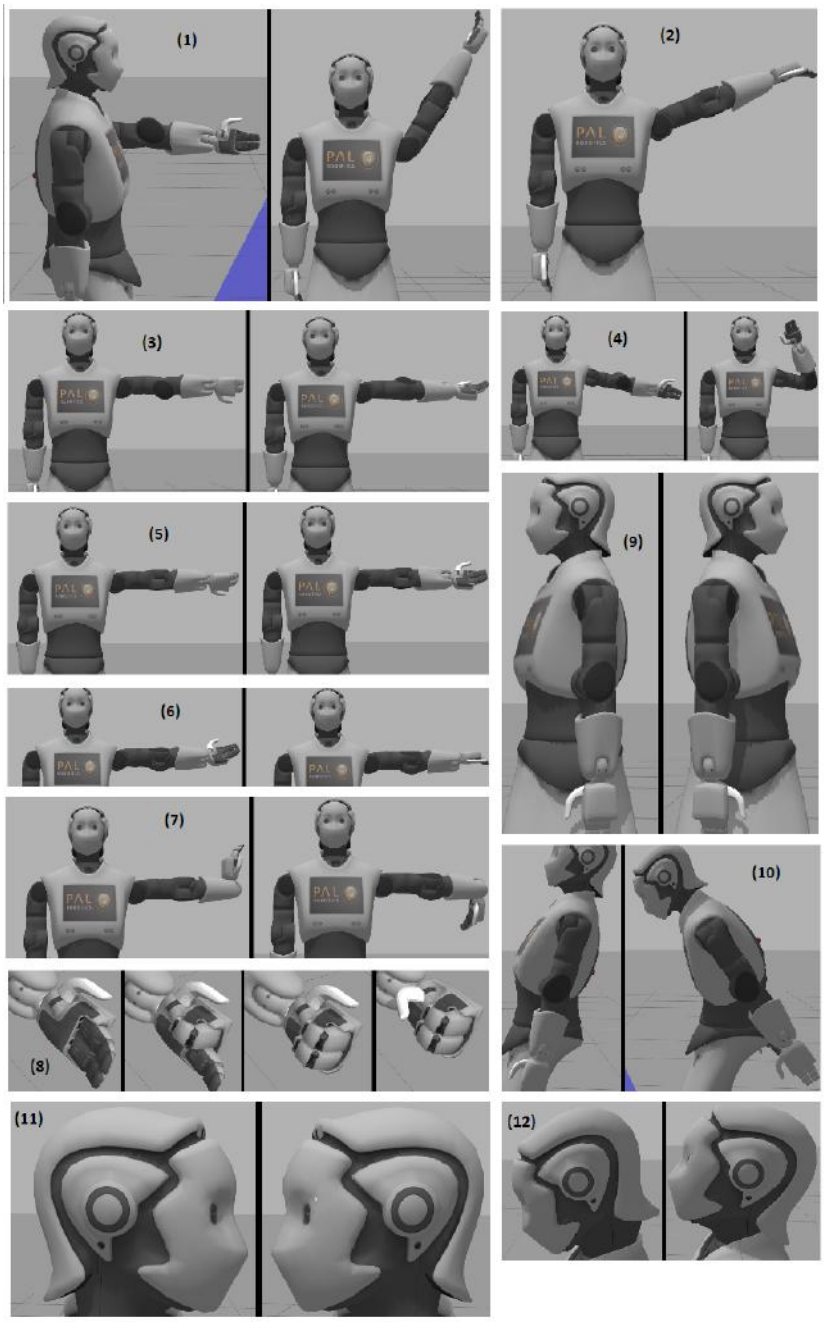


Figura 2.9: Articulaciones del robot REEM.

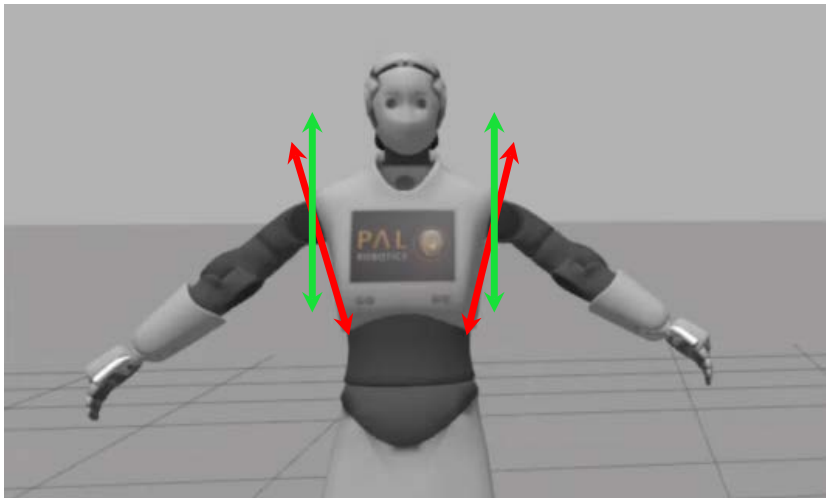


Figura 2.10: Desviación ángulo del hombro.

de gestionar el control robot como tal (navegación, motores, etc). El segundo ordenador se utiliza para la gestión multimedia del robot, destacando en este aspecto que en el pecho dispone de una pantalla táctil de 12 pulgadas con una resolución de 1024 x 768 píxeles que utiliza el gestor de ventanas ligero Openbox. En el presente proyecto, se utilizará esta pantalla para integrar una interfaz gráfica encargada de transmitir comandos al robot y proporcionar un *feedback* al usuario. En cuanto a características audiovisuales, el robot cuenta con 2 cámaras delanteras y una trasera, altavoces, micrófono y dos juegos de leds en las orejas. Estas características, junto con la pantalla táctil facilitan la interacción con el usuario y viceversa.

El robot dispone en los dos ordenadores del sistema Linux Ubuntu 12.04 LTS que junto al fremework robótico ROS gestionan el robot completo. Mas adelante, se detallará el funcionamiento este *framework* utilizado.

Este robot es utilizado en diversos sectores de servicios[7], en eventos, ferias, aeropuertos, seguridad y vigilancia, museos... Algunos ejemplos concretos de uso son como recepcionista para guiar a personas, para entretener, mostrar mapas, llevar pequeños objetos...

[7] https://youtu.be/QZyjCrZj4HM Accedido por última vez 30/03/2016

2.3.3. Robot REEM-C

El robot REEM-C (ver Figura 2.11) es un robot humanoide que nace dos años después que el robot REEM (2012) la misma empresa española Pal Robotics. El robot es algo mas pequeño y ligero que el robot REEM con algunas diferencias notables que se irán comentando a lo largo de la sección.

El robot REEM-C mide 1,65 m y pesa 80 kg. A diferencia del robot REEM, REEM-C cuenta con dos piernas en su parte inferior, que le permiten caminar del mismo modo que una persona. En la parte superior del robot, se puede distinguir la misma distribución que en el robot REEM: cabeza, torso, brazos, muñecas y manos que puede mover gracias a distintos motores. Con estas características, se aprecia que el robot REEM-C es más parecido a un ser humano.

Figura 2.11: Robot REEM-C.

Para poder moverse, el robot cuenta con 44 grados de libertad (10 más que el robot REEM), en la parte superior 32 (7 en cada brazo, 7 en cada mano, 2 en el torso y 2 en la cabeza) y en la parte inferior 6 en cada pierna (a diferencia del robot REEM que contaba únicamente con 2). En este trabajo, al igual que con el robot REEM, el trabajo se ha

centrado en las articulaciones relacionadas con los brazos pero cabe destacar que en las animaciones desarrolladas el robot se mueve con todas las articulaciones, incluidas las piernas. Se puede ver la definición de cada una de las articulaciones superiores en la Figura 2.9 junto con los valores máximo y mínimo de movimiento en la Tabla 2.1. Se hace referencia a las figuras del robot REEM porque las articulaciones superiores (en las que se centrará el desarrollo del proyecto) y los valores máximos y mínimos de las mismas son iguales en ambos robots.

En un primer lugar, se pensó que la morfología del robot REEM-C era diferente a la del robot REEM, ya que a simple vista se puede intuir como el torso del robot REEM-C es mas "cuadrado" y por ello se la posibilidad de que la relación entre el torso y el brazo fuera perpendicular (como en el robot NAO). Dada esta similitud entre los robots, al igual que con el robot REEM, se percibe que el robot cuenta con la misma desviación de la articulación del hombro respecto al torso.

Del mismo modo que el robot REEM, REEM-C cuenta con dos ordenadores integrados aunque mas modernos que los de robot inicial. Uno de los ordenadores es el utilizado como ordenador de control, esto es que se encarga de gestionar el robot como tal (navegación, motores, etc). El segundo ordenador se utiliza para la gestión multimedia del robot, aunque en este modelo, el robot no cuenta con ninguna pantalla táctil. En cuanto a características audiovisuales, el robot cuenta con 2 cámaras delanteras y una trasera, altavoces, micrófono y una matriz de leds 8x8 en la boca.

El robot dispone del sistema Linux Ubuntu 12.04 LTS que junto al *fremework* robótico ROS Hydro gestiona el robot al completo, del mismo modo que el robot REEM. El uso principal que se le está dando a este robot es para investigación. En este campo, está siendo muy útil en la investigación del control al caminar, reconocimiento de objetos y caras, detección de personas, evitar obstáculos, reconocimiento del entorno mediante multi-mapa, navegación autónoma en ambientes atestados, visión estéreo y percepción mediante nube de puntos, interacción Robot-Humano, etc[8].

[8] https://youtu.be/-pEI5-KE2hI Accedido por última vez 30/03/2016

2.3.4. Framework robótico ROS

Sistema Operativo Robótico (en inglés Robot Operating System, ROS) es un *fremework* usado para el desarrollo de software de robots. ROS se creó originalmente en 2007 bajo el nombre de *Switchyard* por el Laboratorio de Inteligencia Artificial de Stanford para dar soporte al proyecto del Robot con Inteligencia Artificial de Stanford (STAIR2), proyecto encargado del desarrollo de robots cualificados para la navegación en entornos de oficinas y la interacción con distintos objetos.

Desde los inicios, el desarrollo continúa en Willow Garage[9], un instituto de investigación robótico, hasta crear en septiembre de 2013 la versión ROS Hydro, utilizada para el desarrollo de este proyecto, y posteriormente ROS Indigo, versión actual del framework y primera versión LTS (*Long Term Support*) del *framework*, que actualmente sigue evolucionando.

Este framework provee los servicios estándar de un sistema operativo, como puede ser abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. La librería está orientada para un sistema UNIX (Ubuntu Linux) aunque también se está adaptando a otros sistemas operativos, aún en estado experimental.

Por tanto, ROS proporciona una colección de librerías y herramientas con el objetivo de facilitar la tarea de crear comportamientos complejos para distintas plataformas robóticas. Hasta el momento, se dispone de multitud de librerías que proporcionan la facilidad de integración con una gran variedad de robots capaces de gestionar diferentes funcionalidades.

[9] http://www.willowgarage.com/ Accedido por última vez 05/04/2016

Básicamente ROS consta de dos partes: la parte que hace las veces de sistema operativo, *ROS*, descrito anteriormente, y *ros-pkg*, una suite de paquetes aportados por la comunidad de usuarios que implementan funcionalidades como localización y mapeo simultáneo, planificación, percepción, simulación, etc.

ROS es completamente de código abierto (BSD) y libre para que todos los usuarios que lo deseen lo utilicen ya que su objetivo principal es permitir a los desarrolladores de software construir aplicaciones de los robots más capaces, rápidas y fáciles de utilizar.

Un ejemplo de uso que es bastante familiar hoy en día es el robot aspirador Roomba. Algunos trabajos [Ruiz et al. 2013] han conseguido conocer la posición de los objetos con ayuda de unos sensores de Kinect y así poder controlar el movimiento para aspirar superficies (Figura 2.12).

Otro ejemplo del uso de ROS es el robot Jaco [Maheu et al. 2011], brazo robótico que puede ser utilizado para gran variedad de tareas comunes que desarrolla el ser humano, como abrir una puerta, coger un vaso y que para algunas personas supone una gran dificultad (Figura 2.12).

Figura 2.12: Robots irobot Roomba (izquierda) y Jaco (derecha).

Los robots REEM y REEM-C utilizados en el presente trabajo son controlados mediante este [Maheu et al. 2011] por lo que se hace fundamental la herramienta para el desarrollo del mismo. En el Capítulo 4 se detallará la integración para su uso con la arquitectura NAOTherapist. Indicar que a diferencia de RoboComp no esta basado en Ice sino en un sistema de intercambio de mensajes.

2.3.5. Simulador Gazebo

Gazebo es un simulador gráfico diseñado para su uso con robots. Es una herramienta de software libre diseñada para probar rápidamente algoritmos, diseño de robots, y probar con los distintos robots utilizando escenarios realistas. Esta herramienta dispone de gráficos de alta calidad, y las buenas interfaces gráficas para facilitar la programación. El simulador tiene distintos motores de físicas que le permiten determinar como interactúan los objetos del entorno entre sí al colisionar.

Gazebo nace en el otoño de 2002 en la Universidad del Sur de California fruto de la necesidad de simular robots en ambientes al aire libre con diversas condiciones. Más tarde, en 2009, John Hsu, un ingeniero senior en Willow, integró por primera vez el framework ROS y el robot Gazebo en el simulador. Desde entonces este simulador se ha convertido en una de las principales herramientas utilizadas en la comunidad de ROS motivo por el cual continúa en desarrollo constante.

El simulador cuenta con múltiples sensores que son capaces de recopilar datos de distintos componentes, como pueden ser sensores de contacto, cámaras, etc. que le permiten simular con bastante precisión las acciones del robot simulado. Algunos de los robots que utilizan este simulador son PR2 (el primero en integrarse), todas las versiones del robot REEM, Pioneer2 DX, TurtleBot, NAO. El propio simulador cuenta con una herramienta que te permite diseñar robots propios únicamente definiendo el esqueleto del mismo.

En la Figura 2.13 se muestran dos ejemplos de la interfaz de Gazebo. En el lado izquierdo se puede ver un entorno de un laberinto, sobre el cual un robot debe planificar un camino, y en el lado derecho, se observa al robot REEM interactuando con algunos objetos.

Durante el desarrollo del trabajo se ha tenido comunicación directa con la empresa distribuidora de los robots pero aunque no ha dado tiempo hacer pruebas con los robots reales. Por este este motivo, el uso del simulador Gazebo ha sido indispensable para

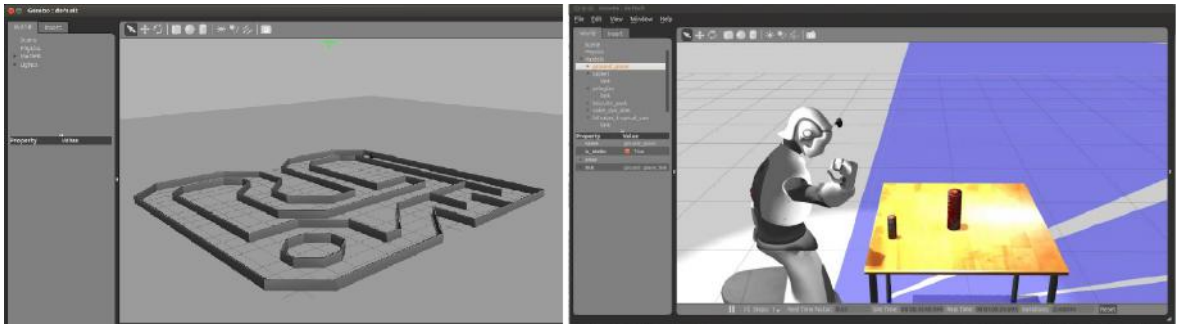


Figura 2.13: Ejemplos de uso del simulador Gazebo.

poder llevar a cabo el desarrollo. Gracias a su buen funcionamiento se puede decir que la integración del sistema con el robot real podría llevarse a cabo correctamente sin dificultad.

2.4. *Retargeting*

Se define el concepto *Retargeting* como la transformación o conversión de un modelo de movimiento a otro. En el ámbito del presente proyecto, es la función que permite convertir los movimientos que de un ser humano para que los reproduzca un robot.

Un caso práctico de este concepto es el trabajo de por Paoulo Sousa [Sousa et al. 2011]. Su estudio se basa en el baile, es decir, propone un método para generar movimientos de baile humanoides transferidos de los datos de captura de movimiento humano. En concreto, se basa en el estilo de baile de la samba. En su trabajo estudia el movimiento de los humanos durante este baile y presenta un método para generar los ángulos para cada articulación del robot, lo que permite la reproducción esta en un robot humanoide NAO simulado.

En el caso de NAOTherapist en el que se ejecutan sesiones de terapia, se utilizan en una serie de poses de las extremidades superiores, en concreto 34 poses, que se han grabado previamente de un esqueleto humano con la Kinect. En este dominio, el

robot enseña al paciente esas poses previamente definidas y posteriormente el paciente debe imitar la postura adoptada por el robot. Además, el robot debe verificar que la postura adoptada por el paciente es la correcta y corregirle en caso de error. Para poder llevar a cabo todo esto, es necesario definir un *retargeting*, que hasta el momento está funcionando en la arquitectura original NOATherapist con el robot NAO. Uno de los objetivos de este proyecto es implementar un *retargeting* propio para los robots REEM y REEM-C. Por todo esto, es necesario analizar el funcionamiento de la visión, el esqueleto y el concepto de *retargeting* para poder comprender la implementación descrita en el Capítulo 4.

Tomando como base un conjunto de ángulos, como son por ejemplo los del esqueleto del cuerpo humano tomados con la Kinect, un *retargeting* es recalcular ese conjunto de ángulos dependiendo del robot que se utiliza para conseguir en él la misma pose que se muestra en el esqueleto inicial, es decir se "traduce" los ángulos del esqueleto humano para que el robot en cuestión imite la misma postura. La función de *retargeting* que se utiliza en este proyecto, usa el esqueleto obtenido de la Kinect para definir la ejecución de poses de los brazos de los robots REEM y REEM-C. Como base de esta función de *retargeting* se ha tomado la función utilizada en la arquitectura NAOTherapist con el robot NAO, que incluye un sistema de visión y calculo de ángulos del esqueleto humano recogido con la Kinect desarrollado en un trabajo anterior [Rossignoli 2015], y toma algunos conceptos de otro trabajo anterior de teleoperación del robot humanoide NAO [Alfaro Ballesteros 2012].

Por tanto, el sistema recoge las posturas y movimientos de los brazos y manos del humano a través del sensor de la Kinect para que el robot imitar dicha postura. En la arquitectura original no se tienen en cuenta los movimientos de las extremidades inferiores a pesar de que el sensor Kinect sí es capaz de reflejarlas, para así evitar una pérdida de equilibrio del robot humanoide. Para este proyecto tampoco serán necesarias las extremidades inferiores ya que el robot REEM-C podría tener los mismos problemas de equilibrio que el NAO y el robot REEM, como ya se ha indicado, carece de piernas.

A continuación se detallan las 3 fases en las que se divide el sistema de la arquitectura NAOTherapist original para hacer el retargeting, ya que esta será la base sobre la que se ha desarrollado el del REEM.

2.4.1. Fase 1: Captura de la postura

En NAOTherapist para obtener la información sobre una imagen en tiempo real, como puede ser la postura de un humano, se utiliza el sensor Kinect que permite capturar imágenes RGB y de profundidad, extrapolando los puntos característicos que forman un cuerpo humano. Esta información se envía continuamente actualizada. En la Figura 2.14 se puede observar una muestra en la que se visualiza la profundidad de una imagen en un instante concreto captada por la Kinect. Esta imagen es la reconstrucción de una superficie 3D que realiza el software oficial de Microsoft a partir de una malla de puntos de profundidad proyectados desde la Kinect.

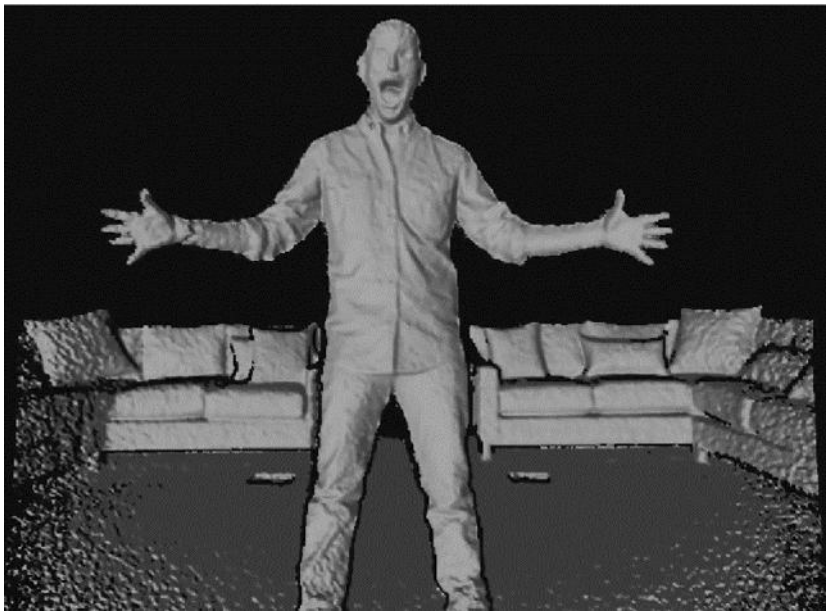


Figura 2.14: Muestra de profundidad de una imagen obtenida con la Kinect.

Para obtener la postura del usuario, se hace uso del componente *Vision* (ver Sección 2.2.1) de la arquitectura original que se encuentra conectado el componente RoboComp *WinKinectComp*. Éste utiliza el software oficial de Microsoft, motivo por el cual está conectado en un equipo con el sistema operativo de Windows y en el se conecta la Kinect. Gracias al sensor y a este componente se obtiene el esqueleto del ser humano con sus puntos característicos del cuerpo en tiempo real. En la Figura 2.15 se muestra el esqueleto recogido con la Kinect con los principales puntos que refleja. Una vez que se tienen los datos de los puntos característicos del cuerpo del usuario, se trata la información en el componente de visión de la arquitectura para conocer la postura o estado del usuario.

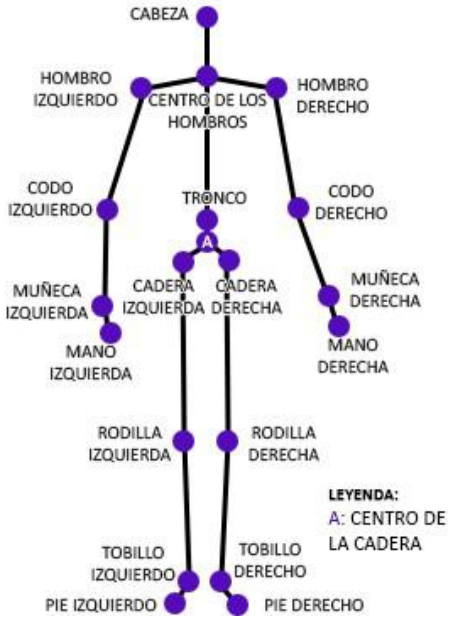


Figura 2.15: Esqueleto humano con los puntos representativos capturados por Kinect.

2.4.2. Fase 2: Cálculo de los ángulos de las articulaciones

Tras haber obtenido el esqueleto del usuario con la Kinect, es necesario calcular los ángulos de las articulaciones del robot usado equivalentes a las obtenidas, es decir convertirlos para poder indicar al robot la pose que debe realizar. Para llevar a cabo esta traducción, la arquitectura NAOTherapist se basa en los planos anatómicos del cuerpo humano, que se describen a continuación y se pueden observar en la Figura 2.16.

- **Plano sagital**: Es el plano que divide el cuerpo verticalmente, es decir es su parte izquierda y parte derecha.
- **Plano transversal**: es el plano que divide el cuerpo en dos horizontalmente, por la cintura, dejando una parte superior y la parte inferior.
- **Plano coronal**: es el plano que divide el cuerpo en parte trasera y delantera.

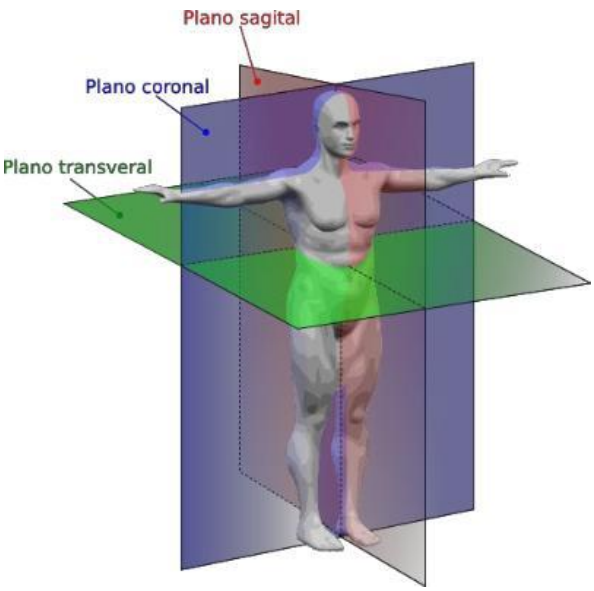


Figura 2.16: Planos anatómicos del cuerpo humano.

Seguidamente se describe el ángulo utilizado para cada una de las articulaciones que usa el robot NAO [Rossignoli 2015] (ver Figura 2.17).

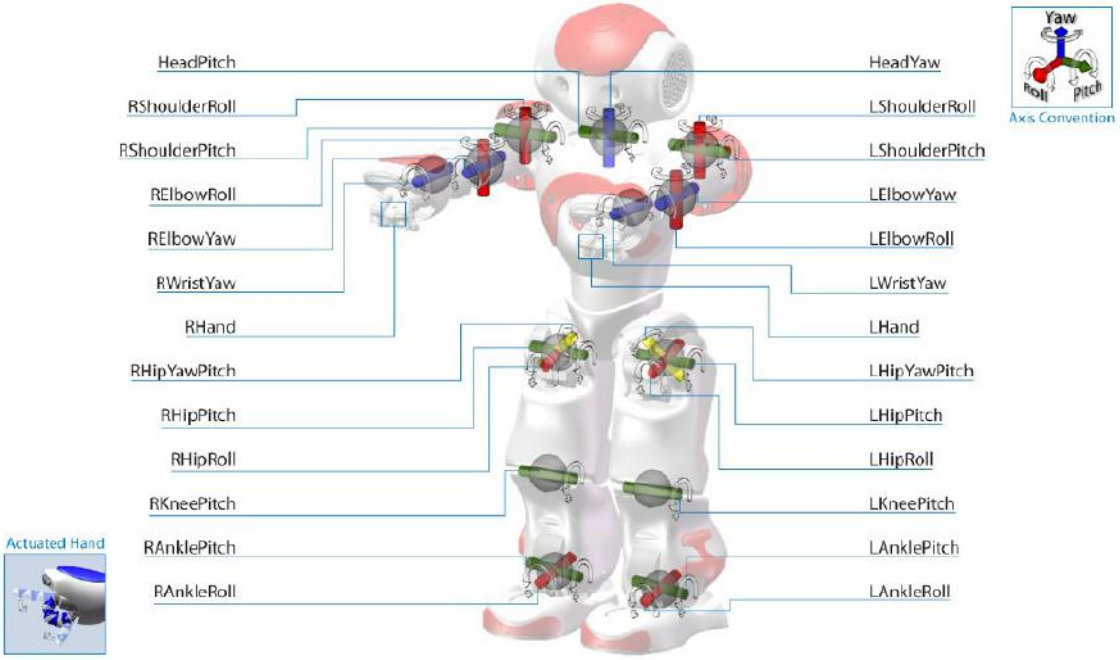


Figura 2.17: Articulaciones del robot NAO.

Ángulo de apertura del hombro (ShoulderRoll)

Para establecer el ángulo de apertura del hombro se usa el plano sagital respecto al brazo, tomando como referencia el seno formado entre los dos vectores.

Ángulo del giro del hombro (ShoulderPitch)

Para el cálculo del ángulo del giro del hombro se usa el ángulo formado por el plano transversal y el brazo, tomando como referencia, al igual que en el caso anterior, el seno entre los dos vectores.

Ángulo de apertura del codo (ElbowRoll)

Para el cálculo del ángulo de apertura del codo se hace uso del ángulo que se forma por la propia morfología del codo, es decir asumiendo que el brazo y antebrazo son los vectores del ángulo, tomando como referencia el coseno de estos.

Ángulo de giro del codo (ElbowYaw)

Para el cálculo del giro del codo es un cálculo más complejo, ya que este ángulo depende del giro del hombro en cada instante. Por ello, se distinguen dos situaciones, dependiendo si el brazo está junto al cuerpo o se encuentra extendido. Este cálculo se basa en un trabajo anterior de teleoperación con el NAO [Alfaro Ballesteros 2012]:

- **Brazo junto al cuerpo**: Para esta situación, se toma como referencia el vector que une los hombros, teniendo esta ángulo un valor de 0º cuando el antebrazo esta girado hacia dentro (hacia el cuerpo), el valor de 90º en el caso de que el antebrazo este girado hacia el frente y el valor de 180º cuando el antebrazo esta girado hacia fuera. Estos valores se pueden ver a modo de ejemplo en la Figura 2.18.

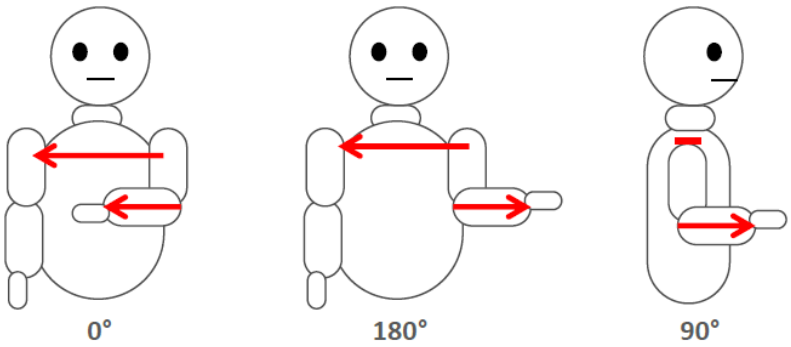


Figura 2.18: Ángulo de giro del codo con el brazo junto al cuerpo.

- **Brazo extendido:** Cuando el brazo se encuentra separado del torso. Para esta situación se tomará como referencia el vector que une el hombro con el lado de la cadera de ese mismo lado. Los valores que se toman para este ángulo son 90º si el antebrazo se encuentra girado hacia arriba (apuntando al techo), el valor 0º si el antebrazo esta girado hacia el frente y -90º si el antebrazo se encuentra girado hacia abajo (apuntando al suelo). Estos ejemplos se pueden ver en la Figura 2.19.

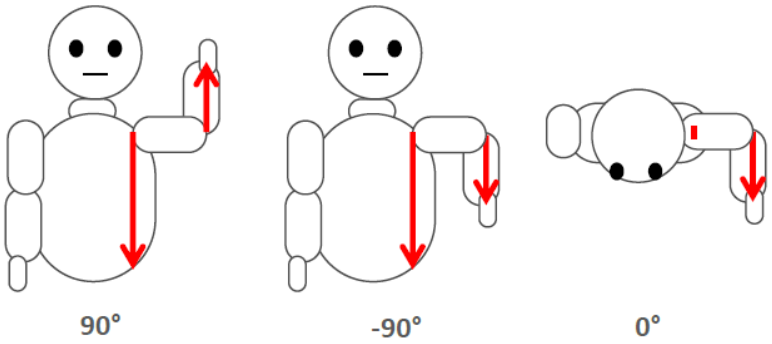


Figura 2.19: Ángulo de giro del codo con brazo extendido.

Teniendo en cuenta estas dos situaciones, se observa que no es posible utilizar un mismo vector para ambos casos, ya que si se usa el vector que une los hombros con el brazo extendido siempre se obtendría un ángulo de 90º. Para que el movimiento sea fluido, se usa la siguiente formula:

$$\frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}})$$

Donde:

- a = Ángulo de giro del codo calculado mediante el vector que une el hombro a la cadera.
- b = Ángulo de giro del codo calculado mediante el vector que une los dos hombros.
- c = Ángulo de apertura del hombro.

Gracias a esta formula es posible que al tener el brazo extendido 90º se anule el sumando, teniendo en cuenta unicamente el ángulo calculado con el vector hombro-cadera. Del modo inverso, se el brazo se encuentra junto al cuerpo (0º), únicamente se tendrá en cuenta el primer sumando que es el correspondiente al vector que une los dos hombros. Para los ángulos intermedios, se hará uso de una ponderación teniendo mayor peso el primer o segundo sumando dependiendo de cuanto este de abierto el hombro. Un ejemplo de esté cálculo se puede observar en la Figura 2.20.

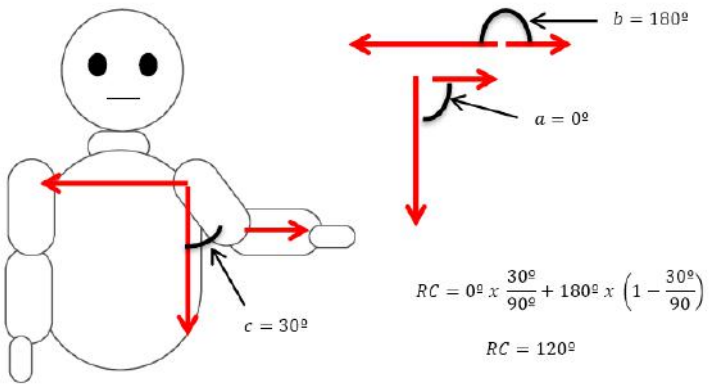


Figura 2.20: Ejemplo de cálculo del giro del codo para el robot NAO.

2.4.3. Fase 3: *Retargeting* original del robot NAO

Finalizado el cálculo de los ángulos de las extremidades superiores es posible que sean necesarias algunas correcciones en ciertos ángulos para afinar la postura que tomará el robot. Entre estas modificaciones se puede invertir algunos ángulos (cambio de signo) o se pueden escalar, ya que es posible que el robot utilizado no use los mismos planos tomados para el cálculo de los ángulos.

El *retargeting* original se creó para el robot NAO en un trabajo desarrollado en el año 2012 en la Universidad Carlos III de Madrid [Alfaro Ballesteros 2012] y se adaptó para poder usarlo en NAOTherapist. Una de las modificaciones en el robot NAO es en el ángulo de la apertura del hombro. En el modelo calculado, la apertura del hombro es de 90 cuando el brazo se encuentra abierto. Sin embargo, en el robot el ángulo de 90º es correcto para el brazo izquierdo, pero para el brazo derecho, esa pose se corresponde con el valor -90º, por ello se hace necesario ajustar el signo del brazo derecho para adaptarlo a los valores deseados para el robot. Para este mismo ángulo, también se escala su valor ya que con el valor calculado la pose que toma el robot es un valor inferior al deseado, es decir, dejando el brazo siempre por debajo de la posición deseada.

Por tanto, en nuestro ámbito, se denomina *retargeting* al cálculo necesario para adaptar el esqueleto humano y el esqueleto de un robot, en el caso de la arquitectura original, al esqueleto del robot NAO. Para el desarrollo de este proyecto, se utiliza un *retargeting* para el robot REEM y REEM-C tomando como base el modelo de ángulos facilitados por el modulo Vision de la arquitectura NAOTherapist y disipando las dificultades encontradas para los nuevos robots, que se detallan en el Capítulo 4 de este documento. Una vez conseguida la adaptación de los ángulos se le envián directamente al robot para que tome la pose deseada.

Capítulo 3

Análisis y diseño del sistema

En este apartado se describirá de forma detallada todo el análisis y diseño del sistema a desarrollar. En primer lugar, se detallan los estándares, las normas y recomendaciones que debe tener el sistema, así como el entono operacional del mismo. A continuación se detallan los distintos tipos de requisitos que debe cumplir el sistema junto con los casos de uso de cada uno.

3.1. Normativa y restricciones del proyecto

En esta sección se explican las normas y restricciones que debe cumplir el sistema. Las **normas** que se deben cumplir son las siguientes:

- Ya que el sistema utiliza datos personales del usuario, es necesario cumplir la Ley Orgánica 15/1999 de Protección de Datos de Carácter Oficial[1].

- Dado que el sistema esta dirigido al uso para la ayuda en terapias con niños, es necesario cumplir la Ley de Orgánica 26/2015 de protección a la infancia y adolescencia[2].

[1] http://www.boe.es/buscar/act.php?id=BOE-A-1999-23750 Accedido por última vez 02/02/2016
[2] https://www.boe.es/diario_boe/txt.php?id=BOE-A-2015-8470 Accedido por última vez

- Los robots utilizados no están específicamente diseñados como equipo médico y por tanto no cumple con los estándares UL o IEC 60601 (o equivalente), sin embargo su aplicación no es obligatoria dado que la interacción que se realiza es sin contacto. En cualquier caso, su uso no está recomendado para menores de 3 años. De 3 a 14 años, su uso debe estar supervisado por un adulto.

Las **restricciones** que se deben cumplir se dividen en restricciones de hardware y software:

- **Restricciones de hardware:**
  - El controlador para captar las imágenes del usuario debe ser el correspondiente a la Kinect XBOX 360.
  - La arquitectura debe ejecutarse en un ordenador con un controlador gráfico NVIDIA para que el simulador Gazebo funcione correctamente.
  - La Kinect, debe estar conectada a un ordenador que que sirve de servidor y gestiona el envío de datos con componente Vision.
  - El ordenador a utilizar para ejecutar el sistema debe tener, por lo menos, 4GB de memoria RAM.

- **Restricciones de software:**
  - El ordenador a utilizar para ejecutar el sistema debe tener instalado el simulador Gazebo.
  - El ordenador a utilizar para ejecutar el sistema debe tener instalado ROS Hydro.
  - El ordenador a utilizar para ejecutar el sistema debe tener instalado un entorno de programación Python.

02/02/2016)

- El ordenador a utilizar para ejecutar el sistema debe tener el sistema operativo Ubuntu 12.04 preferentemente con la interfaz Openbox, ya que es la que utiliza el robot REEM.

3.2. Entorno operacional

En este apartado se define el entorno operacional del proyecto, es decir, se enumeran las herramientas de software y de hardware utilizadas durante el desarrollo del proyecto:

- Ordenador portatil Lenovo Y50-70 con procesador Intel Core I7, 8 GB de memoria RAM, tarjeta gráfica NVIDIA GeForce GTX^{TM} 860M 2GB y sistema operativo Ubuntu 12.04
- Camara Kinect XBOX 360
- Simulador Gazebo junto con ROS
- Paquete Microsoft Office 2013
- Adobe Acrobat XI PRO
- Entorno de programación Python 2.7
- Editor de Latex Online, Sharelatex[3]
- Editor de textos Sublime Text
- Ubuntu Linux 12.04

[3]https://es.sharelatex.com

3.3. Especificación de requisitos

En esta sección se describen los diferentes tipos de de requisitos que se deben tener en cuenta en el presente trabajo. Estos requisitos serán definidos siguiendo el modelo de la Tabla 3.1 donde:

RY - X			
DESCRIPCIÓN:			
PRIORIDAD:	☐ Alta	☐ Media	☐ Baja
NECESIDAD:	☐ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☐ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☐ Alta	☐ Media	☐ Baja

Tabla 3.1: Modelo de definición de requisitos

- **RY - X:** Representa el identificador del requisito, siendo:
  - R: requisito
  - Y: tipo de requisito, requisito funcional (F) o no funcional (NF)
  - X: número entero para identificar el requisito
- **NECESIDAD:** Representa la importancia del requisito descrito en el desarrollo del proyecto. Existen tres niveles de necesidad:
  - Esencial: un requisito es esencial si es imprescindible su cumplimiento para la correcta finalización del proyecto. En caso de no cumplirse, el proyecto no podría llevarse a cabo.
  - Deseable: un requisito es deseable cuando se quiere cumplir, pero a pesar de no conseguirse el proyecto se podrá llevar a cabo con un correcto resultado.
  - Opcional: un requisito opcional es el que puede ser cumplido o no, esto es que no es necesario cumplimentar para terminar el desarrollo del proyecto.

- **PRIORIDAD:** cada requisito de usuaria tendrá definida una prioridad que facilitará la planificación del desarrollador.
- **CLARIDAD:** con esta característica se define el nivel de interpretación que puede tener un requisito. Siendo así tendremos tres niveles para definir la claridad:
  - Alta: un requisito tendrá claridad Alta cuando se pueda dar una única interpretación del mismo. Este es el estado que deben tener todos los requisitos una vez se haya terminado su definición
  - Media: un requisito tendrá claridad Media cuando se pueda dar lugar a pequeñas ambigüedades.
  - Baja: Si el requisito es totalmente ambiguo y por ello puede tener multitud de interpretaciones se le definirá con claridad Baja.
- **VERIFICABILIDAD:** un requisito verificable es aquel del que se puede comprobar si se ha incorporado en el diseño del proyecto.
  - Alta: un requisito facilmente verificable.
  - Media: un requisito tendrá verificabilidad Media cuando es necesario un pequeño análisis para comprobar que se ha incluido en el diseño.
  - Baja: Si el requisito no se puede comprobar si se ha añadido correctamente en el diseño tendrá una verificabilidad Baja.

En las dos siguientes secciones de describen los requisitos siguiendo este modelo y distinguiendo entre requisitos funcionales y requisitos no funcionales. Indicar que al hacer referencia a "El sistema" en los requisitos se hace referencia a "Arquitectura NAOTherapist basada en el *framework* robótico RoboComp".

3.3.1. Requisitos funcionales

A continuación, se detallan los requisitos funcionales siguiendo el modelo descrito. Los requisitos funcionales son aquellos que describen qué debe hacer el sistema desarrollado.

RF - 01			
DESCRIPCIÓN: Si el sistema se encuentra en modo espejo (corrección espejada), el robot realiza la misma posición que reciba del esqueleto de la Kinect.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 02			
DESCRIPCIÓN: El robot reproducirá las palabras indicadas a partir de un texto (*Text-to-Speech*).			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☐ Alta	☒ Media	☐ Baja

RF - 03			
DESCRIPCIÓN: El robot será capaz de reproducir ficheros de audio			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☐ Alta	☒ Media	☐ Baja

Tabla 3.2: Tablas requisitos funcionales del 1 al 3.

RF - 04			
DESCRIPCIÓN: El robot reproducirá animaciones predefinidas con las articulaciones.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 05			
DESCRIPCIÓN: El sistema transforma los ángulos del esqueleto de la Kinect a los ángulos de las articulaciones del robot REEM y REEM-C			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 06			
DESCRIPCIÓN: El sistema dispondrá de una interfaz que simule los leds físicos de los ojos del robot Nao.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 07			
DESCRIPCIÓN: La ejecución de la terapia podrá tener distintos niveles de dificultad con distintas posiciones según sea necesario para cada paciente.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.3: Tablas requisitos funcionales del 4 al 7.

RF - 08			
DESCRIPCIÓN: La reproducción de audio será bloqueante cuando el sistema no continúa su ejecución hasta que el sonido no ha terminado, no bloqueante cuando el audio se reproduce en segundo plano y a la vez continua la ejecución del sistema.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☐ Alta	☒ Media	☐ Baja

RF - 09			
DESCRIPCIÓN: La interfaz tiene 6 botones que simulan los botones físicos del robot NAO original.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 10			
DESCRIPCIÓN: La interfaz será personalizable con diferentes aspectos visuales respecto a colores y formas.			
PRIORIDAD:	☐ Alta	☐ Media	☒ Baja
NECESIDAD:	☐ Esencial	☐ Deseable	☒ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 11			
DESCRIPCIÓN: El robot reproducirá las 34 posiciones que actualmente se utilizan en el proyecto NAOTherapist.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.4: Tablas requisitos funcionales del 8 al 11.

RF - 12			
DESCRIPCIÓN: Si no se establece la conexión con el simulador Gazebo o con el robot real se mostrará al usuario mensaje de error.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 13			
DESCRIPCIÓN: Si no se realiza la conexión con el *framework* ROS se mostrará el correspondiente mensaje de error.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 14			
DESCRIPCIÓN: El robot deberá imitar la postura del usuario en menos de 3 segundos cuando se encuentre en modo espejo.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RF - 15			
DESCRIPCIÓN: Se podrá incluir la secuencia de posiciones para la terapia que desee el terapeuta.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☐ Alta	☒ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.5: Tablas requisitos funcionales del 12 al 15.

3.3.2. Requisitos no funcionales

En esta sección, se detallan los requisitos no funcionales siguiendo el modelo descrito anteriormente. Los requisitos no funcionales son aquellos que describen las restricciones en la implementación del sistema. Es decir, son las propiedades que el sistema debe tener.

RNF - 01			
DESCRIPCIÓN: La arquitectura de referencia NAOTherapist, creada con el framework robótico RoboComp, debe conectar sin disconformidades con el módulo del *framework* robótico ROS que es el que controla el robot REEM.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 02			
DESCRIPCIÓN: El sistema debe conectar correctamente con el simulador Gazebo.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.6: Tablas requisitos no funcionales 1 y 2.

RNF - 03			
DESCRIPCIÓN: El sistema debe ejecutarse en un equipo que disponga un controlador gráfico NVIDIA o Intel.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 04			
DESCRIPCIÓN: La interfaz debe ejecutarse en un hilo de ejecución diferente al del sistema principal.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 05			
DESCRIPCIÓN: Todos los componentes del sistema deben estar en ejecución a excepción del componente *Therapy Designer*.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 06			
DESCRIPCIÓN: Se debe usar la versión del *framework* ROS Hydro.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.7: Tablas requisitos no funcionales del 3 al 6.

RNF - 07			
DESCRIPCIÓN: El sistema se debe ejecutar bajo el sistema operativo Ubuntu 12.04.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 08			
DESCRIPCIÓN: La interfaz gráfica de usuario será compatible con la interfaz de Ubuntu Openbox.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 09			
DESCRIPCIÓN: No se podrán producir colisiones en el movimiento del robot.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☐ Alta	☒ Media	☐ Baja
VERIFICABILIDAD:	☐ Alta	☒ Media	☐ Baja

RNF - 10			
DESCRIPCIÓN: El formato de los ficheros de audio que se desee que reproduzca el robot debe ser *.wav*.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.8: Tablas requisitos no funcionales del 7 al 10.

RNF - 11			
DESCRIPCIÓN: El tamaño de la ventana de la interfaz debe ser de 1024x768.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 12			
DESCRIPCIÓN: Los nuevos componentes se implementarán en el lenguaje de programación Python.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 13			
DESCRIPCIÓN: El nuevo componente desarrollado ReemComp deberá funcionar completamente integrado en la arquitectura NAOTherapist.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 14			
DESCRIPCIÓN: El nuevo componente desarrollado ReemCompC deberá funcionar completamente integrado en la arquitectura NAOTherapist.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.9: Tablas requisitos no funcionales del 11 al 14.

RNF - 15			
DESCRIPCIÓN: El tamaño máximo de las figuras de la interfaz indicará que la posición tomada por el usuario es correcta.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 16			
DESCRIPCIÓN: El tamaño mínimo o no completo (teniendo en cuenta el margen de error) de las figuras de la interfaz indicará que la posición tomada por el usuario es incorrecta.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 17			
DESCRIPCIÓN: Las articulaciones del robot solo podrán tomar un valor establecido entre el máximo y el mínimo especificados según las características de cada uno de los robots.			
PRIORIDAD:	☐ Alta	☒ Media	☐ Baja
NECESIDAD:	☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

RNF - 18			
DESCRIPCIÓN: Antes del inicio de la terapia estará iniciada la interfaz gráfica.			
PRIORIDAD:	☒ Alta	☐ Media	☐ Baja
NECESIDAD:	☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:	☒ Alta	☐ Media	☐ Baja
VERIFICABILIDAD:	☒ Alta	☐ Media	☐ Baja

Tabla 3.10: Tablas requisitos no funcionales del 15 al 18.

3.4. Casos de uso

En esta sección se definen se definen los casos de uso del sistema a desarrollar. Un caso de uso es una descripción de los pasos que debe de seguir un actor (personaje que realiza la acción) para la realización de una tarea concreta. Los casos de uso se pueden definir de forma tabular y de forma gráfica.

En la Sección 3.4.1 se realizará la descripción tabular de los casos de uso y a continuación, en la Sección 3.4.2 se mostrará la descripción gráfica.

3.4.1. Descripción tabular de casos de uso

Para realizar la descripción tabular de los casos de uso se va ha seguir el modelo de la Tabla 3.11. Donde:

CU - X
NOMBRE:
ACTOR:
DESCRIPCIÓN:
PRECONDICIONES:
POSTCONDICIONES:

Tabla 3.11: Modelo casos de uso

- **CU - X**: representa el identificador del caso de uso, siendo:
  - CU: indica que es una tabla de caso de uso
  - X: número natural que enumera los casos de uso.
- **NOMBRE:** nombre que proporciona una pequeña descripción del caso de uso.
- **ACTOR:** tipo de usuario que realiza la tarea.

- **OBJETIVO:** descripción detallada de la finalidad del caso de uso.
- **PRECONDICIONES:** condiciones que deben cumplirse para poder realizar el caso de uso.
- **POSTCONDICIONES:** estado en que queda el sistema tras la realización del caso de uso.

CU - 01
NOMBRE: Detección del usuario
ACTOR: Robot
DESCRIPCIÓN: Comprobar si la arquitectura está reconociendo al usuario correctamente.
PRECONDICIONES: • La arquitectura NAOTherapist debe estar en ejecución. • El sensor de la Kinect debe estar conectado. • El usuario debe estar posicionado enfrente de la Kinect.
POSTCONDICIONES: • En caso de éxito, el sistema continua con su ejecución, indicando con la voz al usuario la siguiente acción a realizar. • En caso de fallo, el robot pregunta al usuario "*¿Dónde estás?*"

Tabla 3.12: Tabla caso de uso 01

CU – 02
NOMBRE: Ejecución de una postura
ACTOR: Robot
DESCRIPCIÓN: El robot realiza una postura que el usuario debe imitar y el robot comprueba que se haya realizado correctamente.
PRECONDICIONES: • El sistema debe estar en ejecución, ROS y Gazebo ejecutados y en espera. • El robot debe indicar una postura. • La interfaz debe estar en ejecución. • El usuario debe estar posicionado enfrente de la Kinect.
POSTCONDICIONES: • En caso de éxito, las figuras de la interfaz se verán con el tamaño máximo y se emite un sonido que indica que la postura del usuario es correcta. • En caso de que la postura del usuario no sea correcta, las figuras de la interfaz se verán con un tamaño entre el mínimo y máximo sin llegar al máximo, y el robot intentará corregir la postura del usuario proporcionándole hasta 3 intentos.

CU – 03
NOMBRE: Descanso
ACTOR: Robot
DESCRIPCIÓN: El robot realiza una pausa en la terapia, durante la misma realiza una animación predefinida para que el usuario se relaje.
PRECONDICIONES: • El sistema debe estar en ejecución, ROS y Gazebo ejecutados y en espera. • El usuario debe estar posicionado enfrente de la Kinect
POSTCONDICIONES: • El robot realiza una animación predefinida como hacer un baile o inspirar/expirar.

CU – 04
NOMBRE: Corregir
ACTOR: Robot
DESCRIPCIÓN: El robot corrige la postura del usuario para que tome la postura correcta.
PRECONDICIONES: • El sistema debe estar en ejecución, ROS y Gazebo ejecutados y en espera. • El usuario debe estar posicionado enfrente de la Kinect. • El usuario debe haber intentado realizar la postura al menos una vez sin éxito.
POSTCONDICIONES: • El robot muestra la postura en la que se encuentra el usuario y después le muestra la postura correcta que debe adoptar para pasar a la siguiente pose.

Tabla 3.13: Tablas casos de uso del 2 al 4.

CU – 05
NOMBRE: Pausar terapia
ACTOR: Terapeuta
DESCRIPCIÓN: El terapeuta indica al robot que debe pausar la terapia
PRECONDICIONES: • El sistema debe estar en ejecución, ROS y Gazebo ejecutados y en espera. • La sesión de terapia tiene que estar en proceso.
POSTCONDICIONES: • Al pulsar el botón en la interfaz de usuario el robot pausará indefinidamente la ejecución de la terapia.

CU – 06
NOMBRE: Saltar pose
ACTOR: Terapeuta
DESCRIPCIÓN: El terapeuta indica al robot que salte a la siguiente pose
PRECONDICIONES: • El sistema debe estar en ejecución, ROS y Gazebo ejecutados y en espera. • La terapia tiene que estar en ejecución.
POSTCONDICIONES: • Al pulsar el botón correspondiente a saltar pose en la interfaz de usuario, el robot omitirá la validación de la postura actual del usuario y pasará al siguiente ejercicio.

Tabla 3.14: Tablas casos de uso del 5 al 7.

3.4.2. Descripción gráfica de casos de uso

En la Figura 3.1 se muestra el diagrama de los casos de uso.

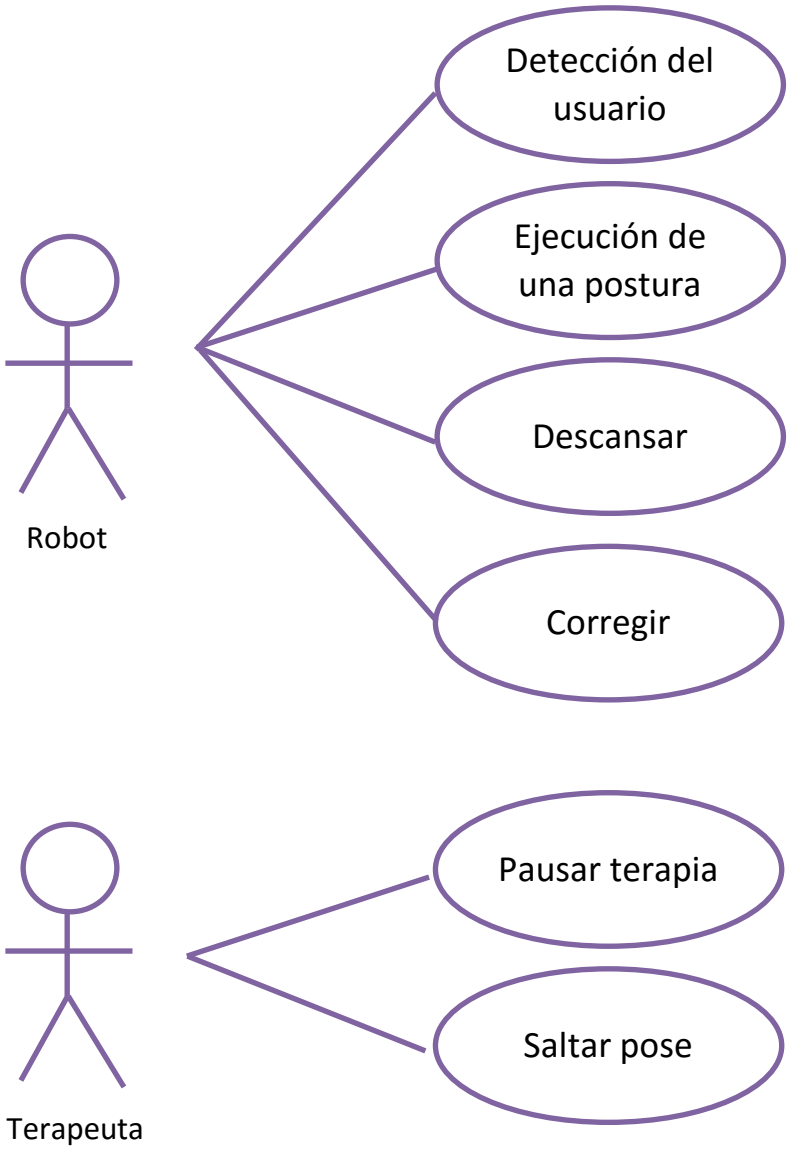


Figura 3.1: Diagrama casos de uso

3.5. Metodología

En esta sección se describe la metodología de desarrollo utilizada. La metodología que se ha seguido para la realización del trabajo es la metodología en cascada, de la que se facilitan mas detalles en la Sección 6.1. Pero además, por la extensión de la parte de desarrollo de este trabajo se ha utilizado para esta fase una metodología basada en prototipos.

Un prototipo es una versión preliminar de un sistema con fines de demostración o evaluación de ciertos requisitos. Sabiendo esto, el modelo de prototipos permite que todo el sistema, o algunas de sus partes, se construyan rápidamente para comprender con facilidad algunos aspectos en los que se asegure que el desarrollador, usuario y cliente estén de acuerdo con lo que se necesita y la solución proporcionada.

De esta forma, este modelo se encarga del desarrollo de diseños para que estos sean analizadas y mejorados a lo largo del desarrollo del proyecto, hasta llegar al resultado deseado. Esta metodología es ideal para medir el alcance del sistema.

Para este proyecto en concreto, se ha aplicado este modelo, ya que es sugerido para proyectos en los que se define un conjunto de objetivos generales, que en este caso el objetivo sería la integración de un nuevo componente en la arquitectura NAOTherapist, pero estos objetivos no están delimitados detalladamente en los requisitos ya que se desconoce el posible alcance del proyecto e incluso el correcto desarrollo del mismo.

El uso de los prototipos implica un método menos formal de desarrollo, donde su fundamento es hacer comprender las especificaciones del producto final. Es este también el motivo de la elección de esta metodología, ya que a lo largo del proyecto se han ido realizando diferentes prototipos que iban encaminando el proyecto hacia los componentes finale REEMComp y REEMCompC, pero sin conocer inicialmente el camino correcto para su desarrollo, en especial con el desarrollo del *retargeting*.

Capítulo 4

Implementación del sistema

En el presente capítulo se describe el proceso de implementación del sistema, comenzando con la integración de las tecnologías necesarias para su desarrollo, detallando las funciones y clases desarrolladas según la arquitectura original. Se realizará un análisis de la función principal del proyecto que es el *retargeting*, encargado de la transformación de los ángulos de las articulaciones. Para finalizar el capítulo, se explicará el dominio utilizado, que en este caso es el mismo que el de la arquitectura original basado en terapias de rehabilitación de extremidades superiores.

La arquitectura del sistema es la que se puede ver en la Figura 4.1, similar a la que se detalló en el Capítulo 2 de este documento. La diferencia radica en el componente de la interfaz del robot NAO, que ha sido sustituido por el del robot REEM o el del robot REEM-C, ya que será posible la utilización de cualquiera de los dos robots.

4.1. Integración de los componentes

El objetivo final de este trabajo es crear dos nuevos componentes con los robots REEM y REEM-C en la arquitectura NAOTherapist [González et al. 2015] y poder intercambiar el robot original NAO. Por ello, como se observa en la Figura 4.1, el

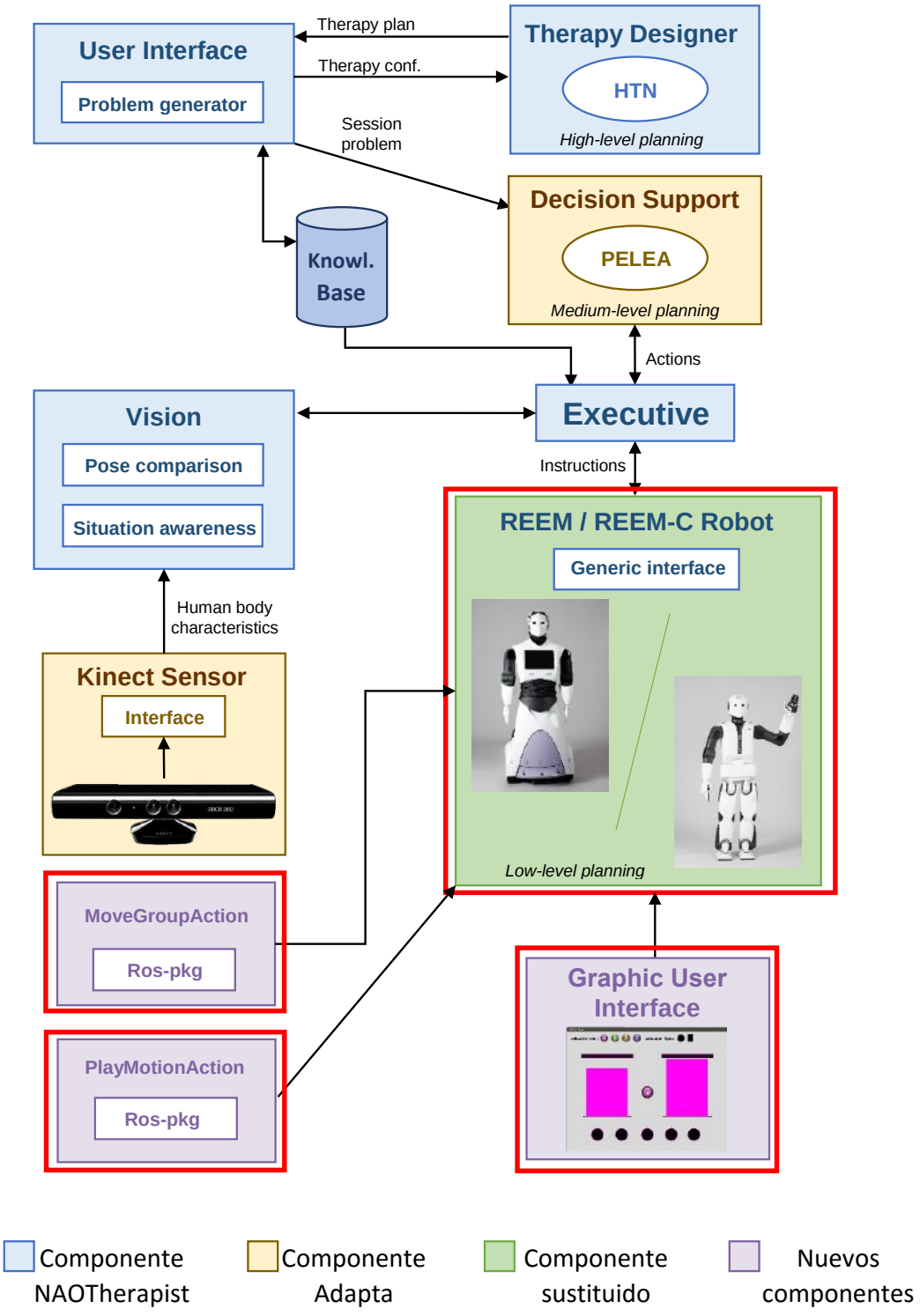


Figura 4.1: Arquitectura del sistema.

objetivo es conservar la mayor parte de la arquitectura original y poder sustituir el robot original por cualquiera de los dos nuevos manteniendo el funcionamiento del robot original NAO. Dada esta situación, es completamente necesario realizar la instalación de toda la arquitectura original NAOTherapist y seguidamente integrar los nuevos componentes desarrollados dentro del *framework* robótico RoboComp para asegurar que no se producen modificaciones de funcionalidad en la arquitectura original. Cabe destacar que la integración de estos tres componentes es común para los dos nuevos robots REEM y REEM-C, por lo tanto, cuando se hace referencia al "robot" en la Secciones 4.1 y 4.2, se entiende que es para cualquiera de los dos.

A continuación se describen las acciones realizadas para la integración de todos los componentes en la arquitectura.

4.1.1. Integración en la arquitectura NAOTherapist original

Para comenzar con la integración de los componentes se ha instalado la arquitectura original NAOTherapist. Para realizar esta integración se han seguido los pasos descritos en el Manual de Instalación (ver Anexo 1). Este manual es un desarrollo propio que resume el proceso seguido asegurando una exitosa integración ya que quedan solventas las dificultades encontradas. La única anotación en cuanto a la arquitectura original, es que en la actualidad está se encuentra integrada en un sistema operativo Ubuntu 14.04 pero dado que para el presente proyecto se requiere de la versión Ubuntu 12.04 (por requerimientos de los robots) se ha realizado la instalación en esta versión también soportada por la arquitectura. Al realizar esta instalación no se ha requerido realizar ninguna modificación adicional para que la arquitectura funcionara correctamente con el robot original NAO.

Dado que en el laboratorio donde se ha desarrollado este trabajo se dispone de varios robots reales, tras la instalación de la arquitectura se ha llevado a cabo la correspondiente comprobación de que dichos robots funcionaban correctamente sobre el entorno que se empezaba a crear para este trabajo. Adicionalmente, se ha compro-

bado también el correcto funcionamiento del sistema en modo simulación, en el caso del robot NAO, utilizando el simulador Choregraphe que es uno de los posibles simuladores para el robot NAO. Para comprobar sobre la arquitectura original el correcto funcionamiento se ha hecho uso de un script de shell que ejecuta todos los componentes necesarios para funcionamiento del sistema.

Una vez instalada la arquitectura original, se comenzó a analizar el modo de desarrollar la funcionalidad para los nuevos robots.

Inicialmente la idea fue ampliar el componente del robot actual NAOComp e integrar en él la funcionalidad de los nuevos robots. Esta idea fue descartada ya que este componente es demasiado dependiente del robot, es decir de la arquitectura de cada robot y las posibilidades de cada uno, por lo que se decidió crear dos nuevos componentes, uno para cada robot (REEMComp y REEMCompC), totalmente independientes del robot NAO de tal forma que la arquitectura siguiera siendo modulable y escalable mas fácilmente. Para que se pudiera ejecutar la arquitectura con los nuevos componentes se ha modificado el script de ejecución original de la arquitectura original de forma que se pueda elegir si se quiere ejecutar la arquitectura con el robot NAO, REEM o REEM-C.

Para establecer la conexión con el resto de la arquitectura es necesario que el nuevo componentes forme parte del *framework* RoboComp, que es utilizado para realizar la comunicación con el componente Ejecutivo y mantendrá el mismo modo de conexión que el componente original, facilitando así la ejecución de varios robots sin modificaciones por parte del usuario final.

4.1.2. Integración ROS

Tal y como se ha detallado en el Capítulo 2, el *framework* róbotico ROS es el encargado de realizar la comunicación entre el nuevo componente y el robot, por lo que ha sido necesaria la integración ROS en la arquitectura. En concreto ROS, permite

enviar instrucciones al robot mediante sentencias de código Python o C++. Para la realización de este proyecto se ha elegido utilizar el lenguaje de programación Python porque a pesar de que el lenguaje C++ es mas rápido, casi todo el resto de la arquitectura esta diseñada en Python y el nuevo componente no tendrá una carga grande de procesamiento. De este modo, se consigue que el nuevo componente siga la línea de diseño de la arquitectura original haciéndola más homogenea.

Gracias a este componente será posible indicar al robot las posiciones o acciones que debe realizar, ya sean posturas simples, como puede ser el robot con los brazos levantados o animaciones, como por ejemplo un saludo. En este trabajo se han utilizado algunas animaciones predefinidas, pero también se ha llevado a cabo el desarrollo de nuevas animaciones que se describen en la Sección 4.3.

A la hora de comenzar el trabajo, no se conocía el funcionamiento interno del propio *framework* ROS, por lo que en un primer lugar se siguierón un conjunto de tutoriales que facilita la página oficial del *framework* ROS[1]. Estos tutoriales explican el funcionamiento básico del *framework*. Se indica la forma de realizar la inicialización del *framework*, modo de creación de los paquetes necesarios y el paso de mensajes entre componentes, que son utilizados en este caso para indicar al robot las ordenes que se desea que realice. Gracias a estos tutoriales se ha comprendido el funcionamiento general del *framework* dotando de los conocimientos necesarios para poder realizar la integración del mismo y así proseguir con la implementación del nuevo componente.

Una vez adquirido el conocimiento acerca del *framework*, se lleva a cabo la integración de ROS con RoboComp (*framework* utilizado en la arquitectura original), requisito indispensable en el trabajo. El resumen de este proceso de investigación e integración del *framework* que finalizo de forma totalmente satisfactoria, se muestra en un manual de instalación que se detalla en la Sección A.2.1 incluido en el manual de instalación del sistema detallado en el **Apéndice A: Manual de instalación** de este mismo documento

[1] http://wiki.ros.org/ROS/Tutorials Accedido por última vez el 01/04/2016

Por último, se ha de comentar que se corría el riesgo de que la integración de ROS en la arquitectura no fuera posible por incompatibilidades con el *framework* RoboComp. El aprendizaje del funcionamiento de ROS y de la propia arquitectura original NAOTherapist ha sido una de las fases mas complicadas del proyecto y que ha llevado bastante tiempo.

4.1.3. Integración Gazebo

PAL Robotics, empresa desarrolladores de los nuevos robots que se van a utilizar en este proyecto, provee un modelo de ambos para facilitar el uso de los mismos en un simulador. Gracias a este modelo cualquier desarrollador puede llevar a cabo las pruebas iniciales de un nuevo programa para cualquier robot de la compañía. Para la realización de este proyecto no ha sido posible tener disponible un robot físico ya que tanto el robot REEM como el robot REEM-C se encuentran en la sede de PAL Robotics en Cataluña. Por lo tanto, para llevar a cabo el desarrollo del sistema se ha utilizado el modelo de la arquitectura de los robots que facilita la empresa usando para la visualización del mismo el simulador Gazebo.

Por tanto, para llevar a cabo el desarrollo del nuevo componente se ha instalado el simulador Gazebo junto con el paquete del modelo del robot REEM y el robot REEM-C para poder visualizar ambos robots como si fueran reales.

Para llevar a cabo la instalación, la misma página oficial de ROS proporciona las instrucciones de instalación de Gazebo y la puesta en marcha de cada uno de los robots (incluye los paquetes descargables de ambos robots). Indicar que a la hora de realizar la integración surgieron algunos problemas con el código propio del robot REEM ya que impedía su correcta instalación. Por suerte, durante la realización del presente proyecto se ha mantenido contacto con los ingenieros de PAL Robotics, que al comunicarles la situación rápidamente actualizaron el repositorio del código y nos facilitaron el mismo, que funcionó sin problemas. Esto no sucedió con el robot REEM-C que se consiguió integrar correctamente sin contratiempos.

Para el correcto funcionamiento del simulador, fue necesario instalar los controladores privativos especiales del controlador gráfico NVIDIA, ya que el simulador no se visualizaba correctamente con el controlador integrado, aunque sí se inicializaba.

Para obtener mas información acerca del simulador y las herramientas y posibilidades de este, se llevaron a cabo alguno de los tutoriales básicos que proporciona la página oficial de Gazebo[2].

En el **Apéndice A: Manual de instalación**, se detalla el resumen de los pasos necesarios para llevar a cabo la integración del simulador Gazebo y el funcionamiento de cada uno de los robots.

Solucionados los problemas técnicos al realizar la instalación de RoboComp y ROS combinados, junto con el simulador y los respectivos paquetes de cada robot, se consiguió tener el entorno completamente preparado para comenzar la investigación del desarrollo del nuevo componente. En la Figura 4.2 se ve cómo Gazebo muestra al robot REEM (arriba) y al robot REEM-C (abajo) una vez se ha iniciado el entorno.

4.1.4. Herramientas para el análisis de los robots y desarrollo del componente

Para poder realizar el desarrollo del nuevo componente y en concreto la integración de las funciones relacionadas con el movimiento del robot REEM y REEM-C (*retargeting*, movimientos simples, animaciones) se ha realizado un estudio de las distintas herramientas disponibles del *framework* ROS y del propio robot que podían ser de utilidad para este trabajo. Tras la investigación, se han obtenido varias herramientas que han sido indispensables para poder llevar cabo el análisis de los nuevos robots. A continuación se detallan las 3 herramientas utilizadas.

[2]http://gazebosim.org/ Accedido por última vez 15/05/2015

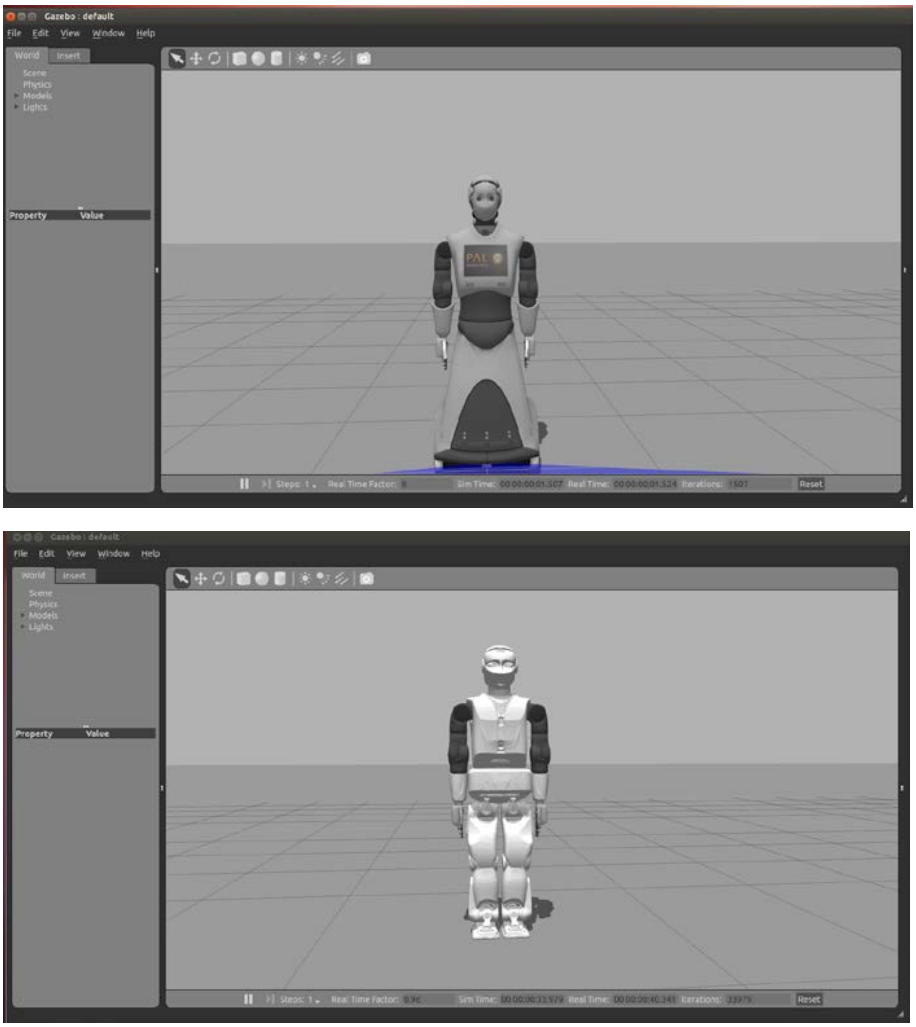


Figura 4.2: Muestra del robot REEM (arriba) y REEM-C (abajo) en el simulador Gazebo.

MoveIt Setup Assistant

Es una interfaz gráfica basada en el *framework* ROS, que permite al usuario configurar cualquier robot. Por tanto, con esta herramienta es posible cargar el modelo del robot y visualizar todas las articulaciones, componentes, uniones, posibles colisiones, diseñar poses y movimientos acordes a las necesidades .

Para entender el funcionamiento de esta herramienta y las posibilidades de la misma, se llevó a cabo una serie de tutoriales, que tomando como ejemplo al robot PR2[3], describían la forma de llevar a cabo la configuración del modelo del robot. Los tutoriales están disponibles en la página oficial de ROS y también en la propia página oficial del componente MoveIt[4].

Gracias a esta herramienta, se estudió fácilmente la configuración y arquitectura del robot. Se observaron las posibilidades de movimiento de cada una de las articulaciones y los límites de estas. Además se observó que para realizar un movimiento, el robot definía un grupo de articulaciones que serían los que se tendrían en cuenta en los distintos movimientos del robot. En la Figura 4.3 se muestra una instantánea de la herramienta descrita durante el análisis del robot REEM.

Por lo tanto, esta herramienta ha sido indispensable para poder conocer la morfología de los robots y obtener la suficiente información para llevar a cabo el desarrollo del *retargeting* de los ángulos del esqueleto de la Kinect a cada uno de ellos. En la Sección 4.3 de este mismo capítulo se facilita más detalle acerca de este desarrollo.

[3] https://www.willowgarage.com/pages/pr2/overview Accedido por última vez 20/03/2016
[4] http://moveit.ros.org/ Accedido por última vez 10/06/2016

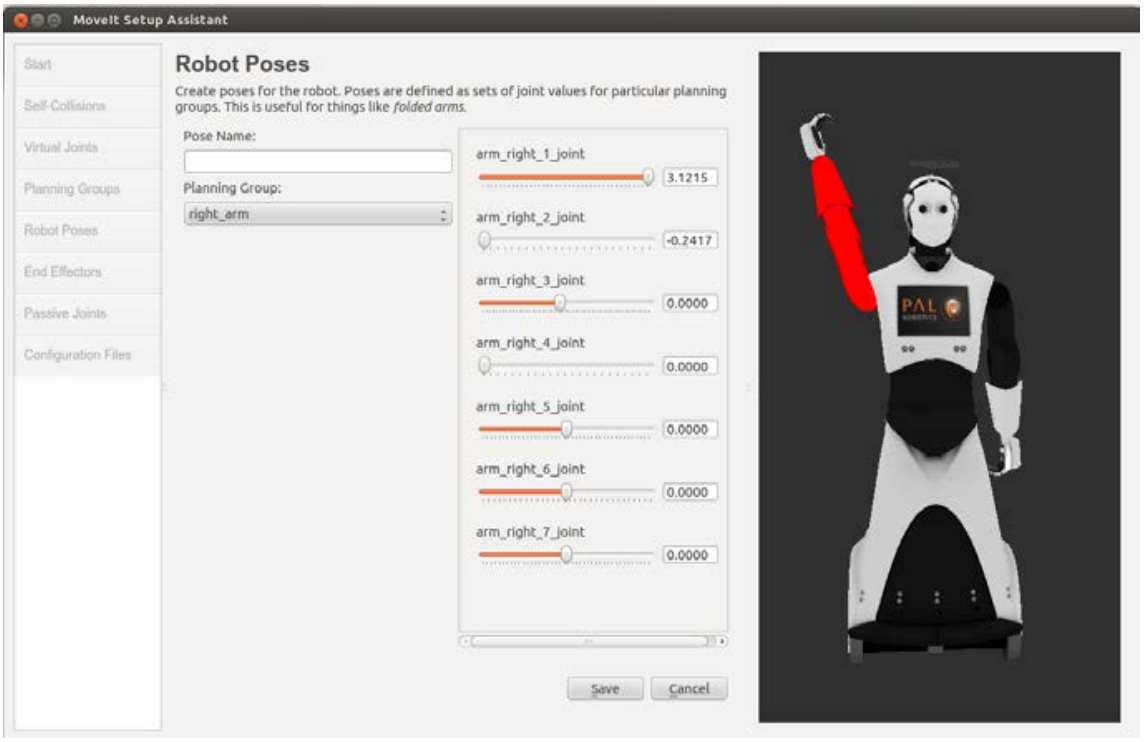


Figura 4.3: Interfaz de MoveIt Setup Assistant.

rqt-joint-trayectory-controll

Esta herramienta nos la facilitó uno de los ingenieros de la empresa PAL Robotics para poder llevar a cabo del diseño de las animaciones. La herramienta, basada en ROS, es similar a la anterior ya que muestra un listado de las articulaciones del robot permitiendo modificar el valor que toman cada uno de sus ángulos. La diferencia con la herramienta MoveIt Setup Assistant, es que es capaz de conectar con el modelo del robot mediante el simulador Gazebo o el robot real, de tal forma que se le puede indicar una postura concreta al robot, se visualiza en el simulador y la herramienta facilita la descripción completa de la postura para poder incluirla en una animación. Por lo tanto, gracias a esta herramienta, ha sido posible llevar a cabo el desarrollo de las animaciones, tomando una a una la descripción de cada una de las poses necesarias para cada animación. En la Figura 4.4 se muestra una imagen de la herramienta.

Figura 4.4: Herramienta para la creación de animaciones.

PlayMotion

Incluido en el propio paquete de cada uno de los robots, integrado con el simulador Gazebo, se encuentra una serie de herramientas y utilidades, entre las que destaca esta que permite ejecutar una postura simple o animación predefinida. Para ello, se ejecuta la herramienta a través de una terminal y se muestra una interfaz gráfica (ver Figura 4.5) en la que se puede indicar el nombre de la postura o animación predefinida para realizar una ejecución de la misma. Gracias al estudio realizado con esta herramienta, se ha obtenido la manera de enviar las instrucciones al robot mediantes código implementado en Python, por lo que es una herramienta clave para el desarrollo de los componente de este proyecto.

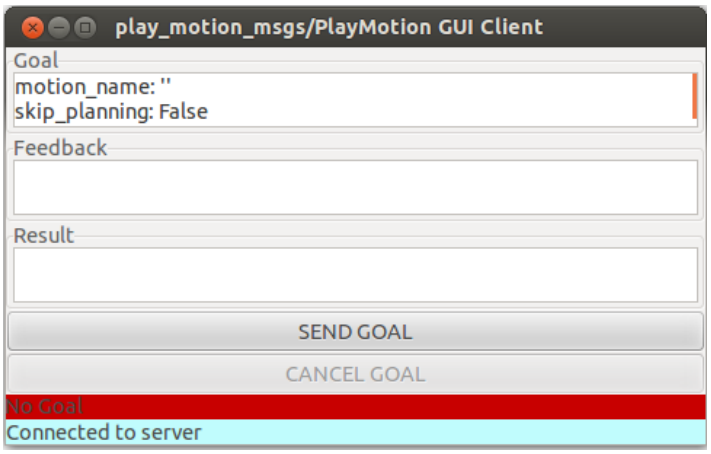


Figura 4.5: Interfaz de PlayMotion.

Indicar que el análisis realizado de estas 3 herramientas y el uso de las mismas ha sido fundamental para el desarrollo e integración de los nuevos componentes en la arquitectura NAOTherapist.

4.2. Establecimiento de postura

Establecido el entorno de la arquitectura NAOTherapist con el nuevo robot, la siguiente tarea era saber de qué forma se envían las ordenes al robot a través del código Python. Investigando sobre las clases del código fuente del robot, se encontró la manera de enviar una postura concreta al robot siguiendo una secuencia de funciones que se ejecutan de forma secuencial. En la Figura 4.6 se muestra esta secuencia que se describe a continuación:

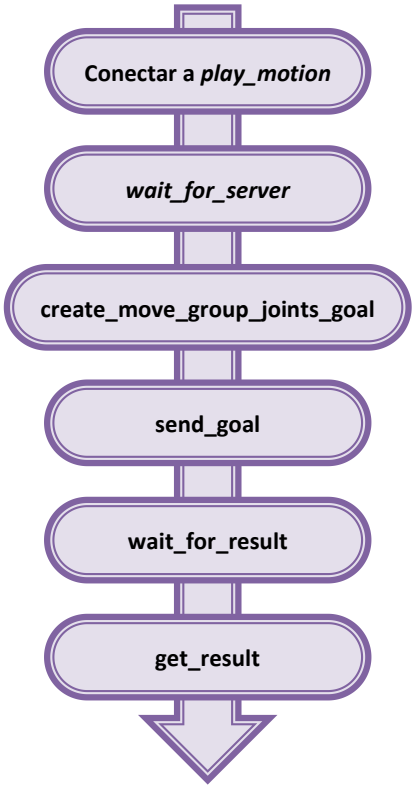


Figura 4.6: Flujo de acciones para la ejecución de una postura.

1. **Conexión a move-group():** al iniciar por primera vez el sistema es necesario crear una conexión con el *framework* ROS que se establece con la acción *actionlib.SimpleActionClient*, lo que permite realizar la conexión con la interfaz del robot que se muestra en el simulador. La conexión establecida permanece abierta por lo que no es necesario volver a realizar esta instrucción cada vez que se desea indicar una acción al robot.

2. **wait-for-server():** Seguidamente, tras establecer la conexión, el *framework* se queda en estado de espera en segundo plano hasta que recibe alguna nueva indicación.

3. **create-move-joints-goal(joint-names, joint-list, group, plan-only):** esta función crea el estado meta que se quiere conseguir, es decir la postura deseada, por lo que recibirá los parámetros necesarios para que el robot pueda realizarla. A continuación se detallan los parámetros requeridos para crear la meta:

   - ***joint-names:*** conjunto de nombres de las articulaciones que se van a mover. Para ambos robots, este conjunto de nombres incluye las articulaciones de los brazos, torso y muñecas del robot.
   - ***joint-list:*** ángulo en radianes que debe tener cada una de las articulaciones para realizar la postura deseada. Esta es una lista de valores que coincide con el tamaño de la lista de *joint-names*, ya que los ángulos son asociados a cada una de las articulaciones definidas.
   - ***group:*** grupo del que forman parte las articulaciones que se van a utilizar en el movimiento. Puede ser un brazo, los dos brazos solos, los dos brazos junto con el torso, o todo el cuerpo.
   - ***plan-only:*** booleano que señala si el robot deberá planificar el movimiento indicado o moverse directamente a la pose indicada. En el caso de planificar el movimiento pasará de la pose actual a la postura indicada teniendo en cuenta las posibles colisiones con su propio cuerpo. Si se diera el caso de que

la pose indicada está en estado de colisión, el robot no realizará el cambio indicado y no realizará dicha postura.

4. **send-goal(goal):** tras crear la meta, se envía al robot mediante el comando *send-goal* para que el robot pueda realizar el movimiento.

5. **wait-for-result():** Al enviar la meta, el componente del robot se queda a la espera de una respuesta por parte del robot que le indique si se ha realizado o no la meta. Es posible indicar como parámetro el tiempo de espera deseado. Pasado el tiempo indicado el sistema continuará con la ejecución independientemente de que se haya alcanzado la meta o no.

6. **get-result():** por último se recibe el resultado de la realización de la pose (meta) indicada, hecho que permite conocer al usuario si esa meta se ha logrado satisfactoriamente o no ha sido realizada.

Siguiendo esta secuencia de acciones se consigue una conexión correcta con el robot y la realización de poses concretas. Para la realización del trabajo se requiere la repetición de esta secuencia para la realización de cada una de las poses requeridas.

Para optimizar el proceso y poder tener en cuenta si una pose se realiza de forma bloqueante o no, en el caso que se desee realizar el movimiento de forma NO bloqueante se ha establecido un tiempo de espera de cero para la función *wait-for-result()*. De este modo, cada una de las metas indicadas se irá guardando en pila y ejecutando en el mismo instante que le corresponda dotando de fluidez al intercambio de posturas del robot. En el caso de que se desee realizar la postura en modo bloqueante, se tendrá en cuenta la espera del resultado de la ejecución de la postura (*wait-for-result()*). Teniendo en cuenta este punto y que las funciones de conexión a (*move-group()*) e indicación de espera del *framework* (*wait-for-server()*) únicamente es necesario realizarlas al comenzar la simulación para la realización de la serie de poses de la terapia, el sistema realiza la conexión con el *framework*, que se queda en espera de las acciones indicadas para indicar la postura. A su vez, cada una de las posturas que se establecen, es una

iteración de las tres funciones que se muestran en la Figura 4.7 (en esta figura se asume que todas las poses se realizan en modo no bloqueante) y que se han descrito en esta misma sección.

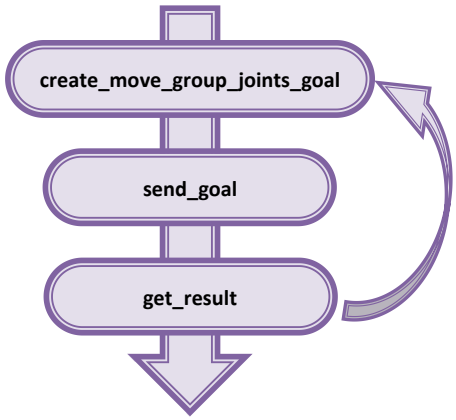


Figura 4.7: Acciones para realizar cada una de las posturas durante la terapia.

Por lo tanto, conocida la forma de establecer una pose y una serie de poses, se ha integrado esta secuencia de acciones en el nuevo componente para poder realizar la ejecución de la sesión de terapia con las distintas poses deseadas.

4.3. Implementación de ReemComp

En la siguiente sección se describe el desarrollo del nuevo componente ReemComp. A partir de este momento, nos centramos únicamente en el desarrollo del componente con el robot REEM ya que la inclusión del robot REEM. Llegados a este punto del proyecto, se tiene la arquitectura NAOTherapist correctamente integrada junto con las tecnologías necesarias para el funcionamiento del robot REEM. Gracias a este nuevo componente se añadirá el robot REEM en la arquitectura NAOTherapist reemplazándolo por el robot original NAO. Una vez incluido el nuevo componente, se realizará la correspondiente evaluación y experimentación sobre el dominio original de terapias ya

integrado en la arquitectura. Es necesario indicar, que este nuevo componente realizar la conexión con el resto de la arquitectura NAOTherapist a través del componente Ejecutivo de la misma.

Para comprender de mejor manera la explicación de esta sección, se ha definido en el Capítulo 2 de este documento las características de este nuevo robot, detallando los grados de libertad del mismo y las articulaciones que se pueden utilizar.

Por tanto en esta sección, se detallará la implementación realizada de del componente, clase que contiene la codificación de las funciones necesarias. También se explicará la clase que contiene la implementación referente a la interfaz gráfica que se comunica con el nuevo componente ReemComp.

4.3.1. Métodos de control

Primeramente, se hace indispensable comprobar el estado del robot, es decir, si el robot se encuentra conectado y si esta en estado de simulación o un entorno real. Por ello, en el trabajo complementario de Carlos Manzano [Manzano Carrasco 2016] se indica la codificación realizada de dos métodos que realizan esta comprobación:

isSimulated():

Este método verifica si el robot REEM está simulado o la ejecución se va a realizar sobre el robot real. El método devolverá el valor **True** si el robot esta simulado, y el valor **False** si se va a ejecutar la terapia con el robot real. Dado que en este proyecto no se ha llegado a realizar las validaciones con el robot real, esta función siempre devolverá el valor de simulación, True.

isConnected():

Este método comprueba que el *framework* esta en ejecución, ya que es imprescindible para que el simulador Gazebo se ejecute y con ello el robot REEM pueda estar disponible para realizar la ejecución de la terapia. Para ejecutar el *framework* ROS se utiliza el comando **roscore**, que contiene los nodos y herramientas necesarias para el

funcionamiento del sistema. *roscore* inicializa la *master*, nodo principal del sistema que permite la ejecución del resto de herramientas y servicios de ROS. Sin esta instrucción, a priori, no es posible la ejecución del resto del sistema. Para comprobar que el robot esta iniciado se utiliza el comando *rostopic*, que comprueba que *topics* están activos. Un *topic* es un bus entre los nodos que pueden intercambiar mensajes, por lo que si entre ellos se tiene el mensaje /**rosout**, esto indica que el *framework* se encuentra inicializado.

El método devolverá el valor **True** en el caso de que el robot se encuentre conectado, y el valor **False** en caso contrario.

Adicionalmente, en el presente proyecto se ha llevado a cabo un análisis de la posibilidad de la correcta/incorrecta conexión del *framework* ROS y el simulador Gazebo independientemente, ya que según la información inicialmente recogida, es necesario realizar la ejecución del *framework* para poder iniciar el simulador. Tras el análisis de esta situación, se ha llegado a la conclusión que debido a que es indispensable la ejecución de la *master* para permitir el inicio de la simulación del robot, el propio simulador Gazebo lleva integrada la *master*. Es decir, el simulador Gazebo al iniciar, realiza primeramente la comprobación de si se encuentra el *framework* en ejecución (la *master*), en caso de ser así, inicia la simulación; pero si se da la situación de no encontrarse iniciado el *framework* de ROS, es el propio Gazebo quien realiza la ejecución de esté para a continuación iniciar el simulador. Con esta información, se observa que para iniciar la simulación del robot únicamente es necesario realizar la llamada al simulador Gazebo con el robot en cuestión, siempre y cuando el sistema se ejecute en modo simulación, como es el caso del presente proyecto.

4.3.2. Reproducción de audio y Text-to-Speech

En lo referente al sonido, se ha utilizado el desarrollo del proyecto de Carlos Manzano [Manzano Carrasco 2016] en el cual se distingue entre la reproducción de un audio a través de un texto y la reproducción de un fichero. Para realizar el control de estas dos situaciones de crean dos métodos **playAudioFile** y **say** comentados brevemente a continuación:

playAudioFile(file,blocking):

Este método es el correspondiente a la reproducción de audio a través de un fichero de sonido que recibe como parámetro junto con la opción de si esté debe ser o no bloqueante en el sistema. Para reproducir el fichero se hace uso de la librería *Pygame.mixer*[5]

say(speech,blocking):

Este método es el correspondiente a la reproducción de audio recibido de un texto, recibiendo esté como parámetro junto con la opción de reproducción bloqueante o no. El método en primer lugar comprueba si existe un fichero de audio que corresponda con el texto indicado y si es así realiza la reproducción de esté del mismo modo que en el método anterior. En caso de no existir el fichero correspondiente al texto indicado, se utiliza un sistema *text-to-Speech* que convierte el texto indicado en audio. El tratamiento de *text-to-Speech* se ha realizado con ayuda del paquete *Pyttsx*[6] que contiene un completo *text-to-Speech* en varios idiomas.

[5] http://www.pygame.org/docs/ref/mixer.html Accedido por última vez el 10/03/2016
[6] https://pyttsx.readthedocs.org/en/latest/# Accedido por última vez el 10/03/2016

4.3.3. Interfaz

El robot NAO dispone de unos leds en los ojos que en la arquitectura original NAOTherapist se utilizan para facilitar al usuario un *feedback* de la postura que esta realizando, así cuanto mas se acerque la postura tomada por el usuario a lo postura correcta el robot mostrará en sus ojos una intensidad mayor del color verde. Además con estos leds en los ojos, es posible simular que el robot se encuentra con los ojos cerrados o incluir distintas animaciones visuales con ellos. Adicionalmente, el robot NAO, cuenta con 6 botones distribuidos en distintas partes de su arquitectura, así tiene 3 botones en la cabeza, 1 en cada pie y uno central en el pecho. Algunos de estos botones son utilizados en la arquitectura original para poder facilitar la interacción del terapeuta con el robot. De modo que por ejemplo los botones de la cabeza permiten pausar la terapia o incluso pasar, dar como correcta, una postura que un paciente no es capaz de realizar en un determinado momento.

Para este proyecto era necesario incluir toda esta funcionalidad de interacción y *feedback* al usuario pero robot REEM no tiene botones y los leds que tieneen los laterales de las orejas no son demasiado visibles ni intuitivos para dar el *feedback* al usuario.

Dado que el robot REEM cuenta con una pantalla táctil en el pecho que se utiliza únicamente para mostrar contenido multimedia, se decidió crear una interfaz gráfica que tuviera la funcionalidad de los leds y los botones. Así en el robot REEM se podría mostrar la interfaz en el pecho del robot, del mismo modo que se visualiza en la Figura 4.8.

Indicar que durante el desarrollo del proyecto se diseño un primer prototipo en el que se mostraban los botones y unos ojos que cambiaban de color del mismo modo que la arquitectura original. Adicionalmente para el presente proyecto se incluyeron una serie de mejoras que se describen en la Sección 4.3.3.2. En las siguientes secciones se describe el proceso de implementación de la interfaz junto con las mejoras añadidas.

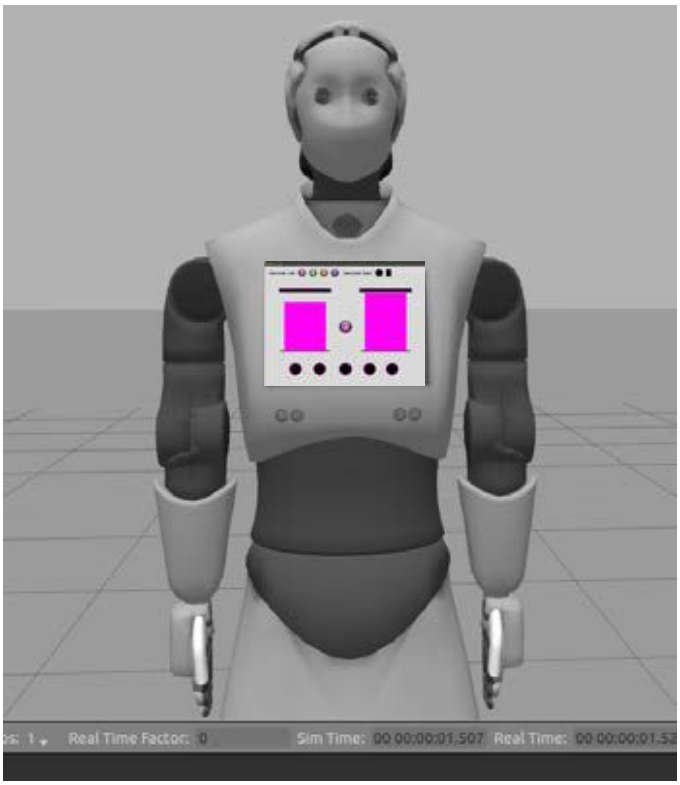


Figura 4.8: Robot REEM con ejemplo de interfaz en la pantalla tactil.

4.3.3.1. Primer prototipo

La interfaz, al igual que el resto del sistema ha sido implementada en lenguaje Python, haciendo uso para su creación de la librería *Tkinter*[7]. Esta libreria es un paquete estándar en al ambito de las interfaces gráficas de usuario, bastante ligera y muy utilizada. Se ha elegido esta herramienta por la buena documentación de la misma, así como por la sencillez para la creación de la interfaz requerida. Además, una ventaja del uso de esta librería es su compatibilidad con *Openbox*[8], gestor de ventanas en entornos UNIX, diseñado para consumir los mínimos recursos e integrado en el ordenador multimedia que dispone el robot REEM en el pecho del mismo. Asegurando

[7] https://wiki.python.org/moin/TkInter Accedido por última vez 12/06/2016
[8] http://openbox.org/wiki/MainPageAccedido por última vez 12/06/2016

esta compatibilidad es seguro que se pueda ejecutar la misma en el robot REEM.

En primer lugar se ha credo una ventana sin contenido de un tamaño 1024x768 ya que es este el tamaño de la pantalla integrada en el robot REEM de tal modo que se muestre a tamaño completo la interfaz en el pecho de dicho robot, además, en otros dispositivos, al ser un tamaño bastante estándar se visualizará correctamente. Esta ventana es la base del diseño de la interfaz.

Como ya se ha indicado, inicialmente se diseño un prototipo base que se quería fuera lo mas semejante posible a la funcionalidad y aspecto que se muestra en la arquitectura original. De esta forma, se incluyo en la ventana vacía unos ojos y los 6 botones disponibles del robot NAO. En la Figura 4.9 se muestra una imagen de este primer prototipo diseñado.

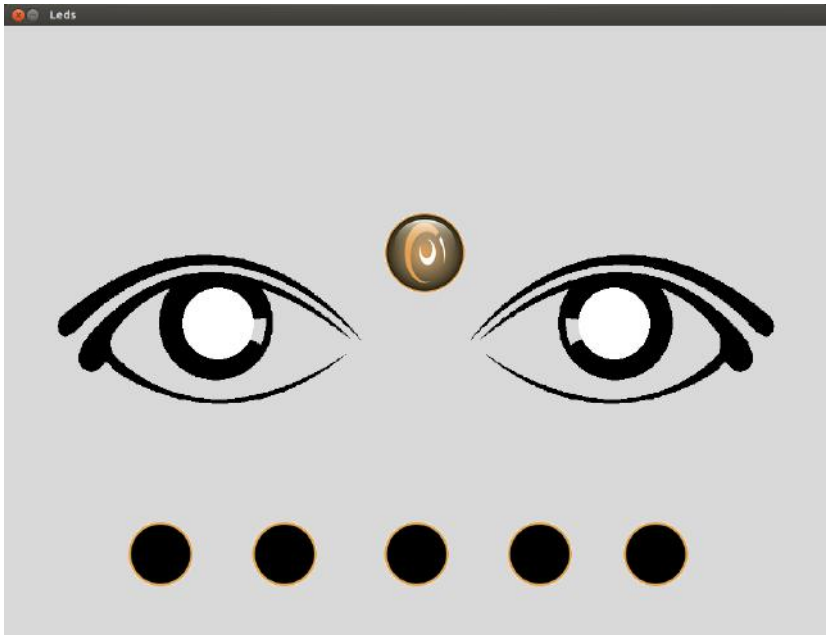


Figura 4.9: Interfaz inicial completa.

Ojos

En la Figura 4.9 se puede ver un ejemplo de los ojos incluidos en la interfaz inicial. Para ello se han utilizado dos imagenes, una para cada ojo, en formato gif, para que el interior de cada ojo se pudiera añadir una figura circular que muestre el color que se indique en la arquitectura del mismo modo que los leds de los ojos del robot NAO original. Para cualquiera de los objetos que se han añadido en la interfaz se han usado medidas relativas, de modo que si se redimensiona el tamaño de la ventana esto no afecta a la estructura de la interfaz.

Una vez incluido el diseño de la parte de los ojos, es necesario conectar con el componente del robot, que al mismo tiempo se conecta con el resto de la arquitectura. Para esto, se inicializa la ventana vacía en el componente del robot y se importa la clase creada de la interfaz para permitir el acceso a las funciones desarrolladas de la misma, de forma que se pueden dibujar los elementos incluidos.

Una vez se ha inicializado la interfaz, ésta se queda en espera, en segundo plano, comprobando continuamente si se requiere alguna acción sobre la misma. Como por ejemplo cambio de color de los ojos, la pulsación de un botón,... Esto ha sido un impedimento, ya que tanto el componente del robot y como interfaz se quedaban en espera de recibir instrucciones por lo que, al ser ejecutados ambos en un mismo componente, se producía una situación de bloqueo que impedía la ejecución de la interfaz o del componente del robot. Para solventar esta situación, se ha decidido inicializar la interfaz en el componente del robot en un hilo de ejecución diferente, de modo que ambas ejecuciones pudieran seguir su curso sin producir el bloqueo de ninguna de las partes.

setLeds(ledGroup,intensity)

Integrada la interfaz, el componente del robot utiliza el método *setLeds* para recibir del componente Ejecutivo el color que se desea mostrar en la interfaz. Esté recibe como parámetros el conjunto de leds que se desea modificar (*AllLeds, LeftFaceLedsRed,LeftFaceLedsGreen, LeftFaceLedsBlue, RightFaceLedsRed, RightFaceLedsGreen, RightFaceLedsBlue*), junto con la intensidad de color (de 0 a 1) de ese grupo.

Realmente cada grupo, a excepción de *AllLed* corresponde a uno de los valores RGB que forman un color para cada ojo. Así el ojo derecho tendrá 3 grupos que se corresponden con la intensidad de rojo, verde y azul que se juntan para formar el color que se desea mostrar en ese ojo, y del mismo modo se estructura en el ojo izquierdo. Se definen como grupo porque en el robot original, cada grupo contiene 8 leds que definen cada uno de los colores RGB (rojo, verde y azul) y que se muestran con mayor o menor potencia dependiendo de la intensidad que se le indique.

La intensidad se recibe como un número normalizado entre los valores 0 y 1, ya que así lo utiliza el robot NAO. El valor correcto para mostrar en la interfaz, es necesario que esté en formato RGB, es decir comprendido entre 0 y 255, por lo que se realiza una conversión para adaptar el valor en el nuevo rango. Para mostrar esta funcionalidad más claramente, a continuación se detalla un ejemplo:

Se tiene una intensidad de 0 para los grupos *RightFaceLedsRed, RightFaceLedsGreen, RightFaceLedsBlue* por lo que el color que se muestra en el ojo derecho es el negro. Si se realiza la siguiente llamada del método: *setLeds(RightFaceLedsGreen,1)*, se cambiará el valor del grupo del color rojo derecho a 1, que al convertir el valor al rango del formato RGB será 255. Los grupos de leds guardan el último valor asignado, de forma que si se realiza una modificación sobre un grupo, está no afecta al valor de los demás. Esto no se cumple para el grupo *AllLeds* ya que este realiza la modificación del valor para todos los grupos. Por lo tanto, el color resultante en formato RGB será *0 255 0*, por lo que del color negro, el ojo cambiará a ser verde.

Una vez dispuestos los valores en formato RGB, se invoca al método *updateSetLedsColor* de la interfaz que recibe el array con los valores para cada grupo y los círculos de los ojos, simulando el cambio de los ojos del robot NAO. El resultado final del ejemplo anterior se puede observar en la Figura 4.10

Como ya se ha indicado en los objetivos del proyecto, se ha desarrollado una mejora de la interfaz que incluye una visión diferente de estos componentes que se describe en la Sección 4.3.3.2.

Figura 4.10: Ejemplo de la simulación de los leds de los ojos.

Botones

Como ya se ha mencionado, el robot original NAO, tiene 6 botones físicos: 1 en el pecho (para encender/apagar el robot), 1 en cada pie y 3 en la cabeza. Del mismo modo, se han añadido estos 6 botones en la interfaz: uno central con el logo de la empresa PAL Robotics que corresponde con el botón del pecho, y 5 en la parte inferior, que son los correspondientes al resto de botones indicados. En la Figura 4.11 se puede ver la equivalencia de los botones del robot NAO en la interfaz inicial creada. En cuanto al funcionamiento, se utiliza el método *getLastButtonPressed*:

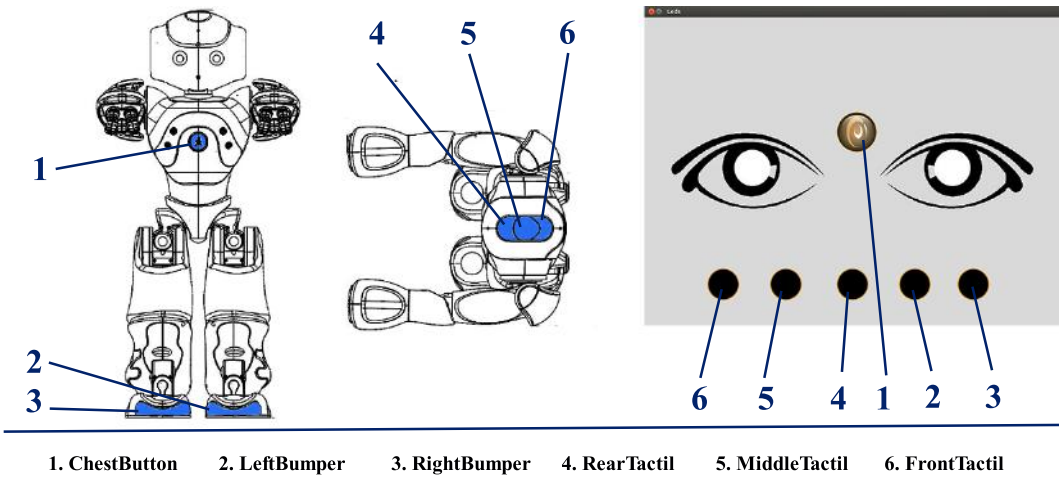


Figura 4.11: Equivalencia entre botones de NAO y REEM.

getLastButtonPressed():

Este método guarda el nombre del último botón pulsado y lo devuelve. La funcionalidad asociada a cada uno de los botones esta implementada en el componente Ejecutivo de la arquitectura original por lo que no se ha tenido que realizar ningún desarrollo adicional para estos botones. En la interfaz cada vez que se presiona un botón se actualiza la variable que guarda el último botón pulsado y esta de devuelve al componente Ejecutivo cada vez que la solicite.

Diseños (Skins)

Finalmente, en este primer prototipo se ha añadió la posibilidad de personalizar la interfaz con diferentes *skins* para facilitar la interacción en función de los requerimientos del terapeuta o los gustos del paciente.

Figura 4.12: A la izquierda skin cara de niña, a la derecha skin cara de médico.

El cambio entre las diferentes skins de la interfaz inicial es totalmente manual, es decir el terapeuta debe elegir cual de las skins desea que se visualice cada paciente antes de realizar la terapia. En la Figura 4.12 se muestran los primeros diseños para el cambio de interfaz realizados. Con estos dos diseños se da valor a la interfaz, permitiendo dar un ambiente mas amigable para la terapia. Así como se podía semejar a la cara de una

niña o la cara de un médico, dejando la skin original que se muestra en la Figura 4.9 para entornos mas serios, en la que se eligió el color naranja por seguir la línea de colores de la compañía creadora de los robots.

Igualmente, estos diseños han sido mejorados al realizar la mejora de la interfaz con las nuevas interfaces que se describe en la siguiente sección.

4.3.3.2. Interfaz mejorada

Con el prototipo inicial funcionando correctamente, se observó que no era lógico ni intuitivo visualizar unos ojos adicionales en el pecho de un robot o en una pantalla anexa, por lo que se tomó la decisión de mejorar el diseño de la interfaz para que *feedback* facilitado fuera más claro y tuviera mayor sentido con los nuevos robots.

Mejora del diseño

Para ello, se decidió cambiar la figura de los ojos por unas barras que cambiaran de tamaño según fuera más exacta la postura tomada por el usuario, de modo que alcanzado el tamaño máximo de la misma significaría que la postura es totalmente correcta y un tamaño mínimo o sin alcanzar un cierto umbral indicaría una postura incorrecta.

Por tanto se eliminaron las imágenes de los ojos y se incluyeron 3 rectángulos por cada ojo. Un rectángulo inferior que sirviera de base, es decir de punto de partida para la comprobación de la pose tomada; un rectángulo bastante superior, que fuera el punto límite, máximo, que puede tomar la figura para indicar la correcta postura. Por último, un gran rectángulo central que es el que cambia de tamaño, empezando siempre desde el rectángulo inicial y creciendo hacia el rectángulo superior (ver Figura 4.13).

Para realizar este cambio, se han reutilizado las funciones del prototipo, incluyendo las correspondientes modificaciones para adaptar la nueva forma de actualización de los objetos que facilitan el *feedback*. Para ello en este diseño, se establece un color fijo a las figuras (siendo más claro el color del rectángulo central) para que no distraiga al

usuario y que únicamente se centre en el tamaño que toma la figura.

Los valores recibidos para facilitar el *feedback* seguían siendo los grupos la intensidad para cada grupo de leds, que a priori no servía para establecer el tamaño de una figura. Analizando nuevamente los valores recibidos del componente Ejecutivo, al comprobar la postura se observa que el valor del grupo verde para el lado izquierdo y derecho siempre es 1 y son los otros dos grupos los que van cambiando su valor para modificar la intensidad del color verde en el diseño original y así en el nuevo mostrar un gradiente de tamaño de la figura. Con esta información, y viendo que los valores del grupo rojo y azul siempre son iguales, se ha implementado una función de conversión que transforma el valor de la intensidad recibida del grupo rojo del lado izquierdo y derecho en el valor que se debe establecer en el rectángulo central. Con esto, se ha obtenido el resultado deseado que es visualizar dos rectángulos centrales (uno que corresponde al lado izquierdo y otro que corresponde al lado derecho) que muestra al usuario/paciente lo que le falta para establecer la postura correcta.

Para llevar a cabo este cambio, se ha tenido en cuenta el margen de error establecido en el que se asume una postura como correcta. Este margen en la arquitectura original tiene en cuenta la dificultad de obtener el 100 % de la postura correcta ya que en algunas posturas es posible que se pierda precisión en el traspaso de las imágenes de la Kinect y el *retargeting* posterior. Con el nuevo diseño, si la postura no es 100 % correcta, se visualiza una pequeña franja blanca que indica lo que falta para llegar al 100 %. Incluso si la postura bien hecha, es posible que no se visualizara de dicha manera en la interfaz. En el diseño original y con el robot NAO, no se daba esta situación dado que dado que la intensidad de un color es un parámetro más difícil de distinguir entre niveles semejantes. Por este motivo se amplió el ancho del rectángulo superior, para poder tener en cuenta este error, o mejor dicho, mostrar más exactamente el momento en el que la postura comienza a ser correcta. Este valor de error viene definido por la arquitectura NAOTherapist original en un valor de 0,2 establecido en un rango de 0 a 1. En la Figura 4.13 se muestra el nuevo diseño de la interfaz.

Para tener en cuenta los cambios de color de las animaciones, al realizar la modificación se ha mantenido el cambio de color de las figuras para las animaciones o acciones no incluidas expresamente en la realización de la terapia. De este modo por ejemplo, para simular el cierre de los ojos del robot en el nuevo diseño se muestra la figura central en color negro, del mismo modo que en el diseño inicial. Las animaciones que tienen en cuenta este cambio de color se detallan en la Sección 4.3.4. En la arquitectura NAOTherapist tiene un pequeño fallo de generalización al considerar los "leds" como "*feedback* de la pose", por lo que abrir y cerrar los ojos u otras animaciones con colores dificultan el desarrollo de esta interfaz. En NAOTHerapist se está trabajando para solventar este problema de diseño y separar animaciones de la información de *feedback* de postura.

Intercambio de skin

Adicionalmente, se ha creado un diseño con la misma funcionalidad y estructura pero con círculos en lugar de barras. Del mismo modo, se tiene un círculo mas grande y un punto que corresponden con el rectángulo superior e inferior; junto con un círculo central que es en el que se muestra el estado de la correcta postura tomada por el usuario.

Llegado a este punto, se tenían dos interfaces distintas por lo que se han implementado nuevos botones, que permiten el intercambio de ellas al inicio de la terapia o en tiempo de ejecución, para que pueda ser el propio paciente quien indique con qué interfaz desea realizar la terapia. Para el funcionamiento de los nuevos botones se ha incluido la nueva funcionalidad en el método *getLastButtonPressed*, de modo que en el momento en que recibe que el último botón pulsado corresponde con uno de los botones de cambio de la interfaz realiza una llamada al método *updateSkin(color,figura)* que es el encargado de redibujar la interfaz con la nueva figura. Dada esta nueva funcionalidad es trivial incluir por ejemplo 3 colores adicionales que igualmente se pudieran cambiar en tiempo de ejecución.

Por lo tanto, finalmente se cuenta con dos modelos finales de interfaz que se

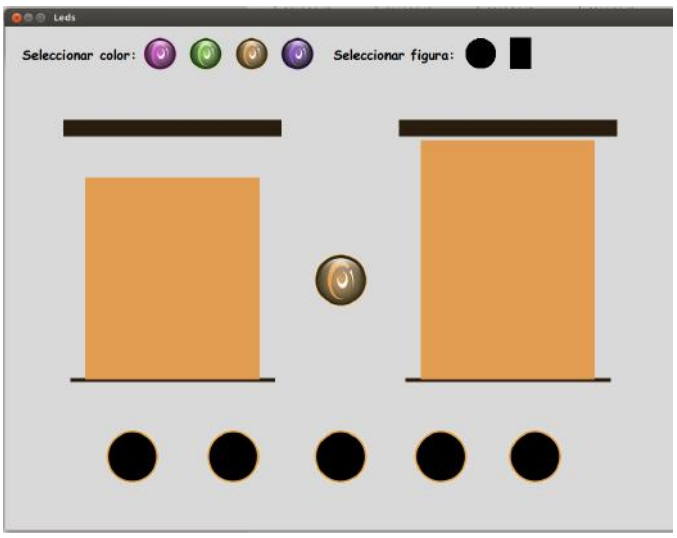


Figura 4.13: Modelo final nueva interfaz (barras).

pueden cambiar entre 4 colores distintos. En la Figura 4.13 se muestra un ejemplo del resultado final de la interfaz, en el que se ven los nuevos botones de cambio de skin, mostrando un círculo de color con el logo de la empresa PAL Robotics para cada uno de los colores disponibles y un rectángulo y círculo negro para poder elegir la figura de interfaz deseada. Adicionalmente en la Figura 4.14 se pueden ver algunos de los modelos disponibles para cambiar el diseño de la interfaz en tiempo de ejecución.

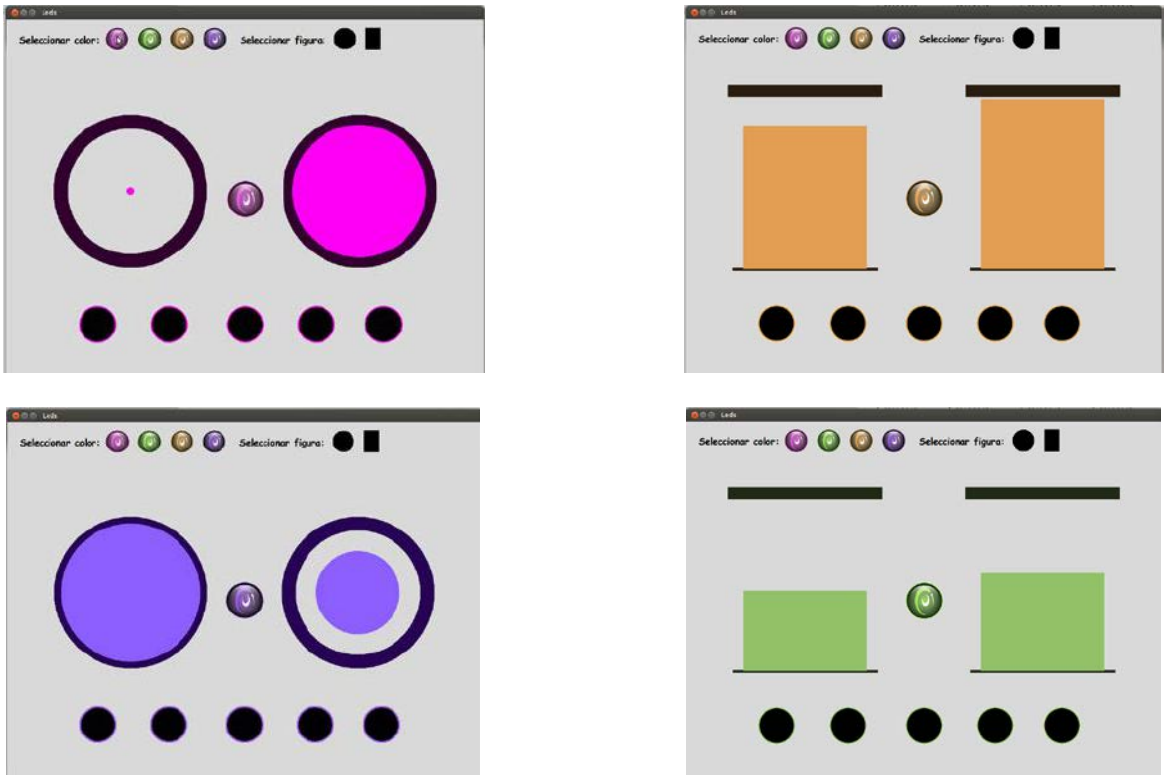


Figura 4.14: Muestra de modelos para el cambio de skins

4.3.4. Animaciones

Para favorecer la interacción con el usuario se han integrado en la arquitectura una serie de animaciones predefinidas. Tanto el robot REEM como el robot REEM-C ya tenían integradas una serie de animaciones como por ejemplo en la que el robot saluda, pero se ha completado este listado con animaciones extra que son utilizadas en la arquitectura original. Para la ejecución de las animaciones se usa la librería *play-motion* (detallada en la Sección 4.1). En la Figura 4.15 se muestra un esquema de las acciones seguidos para realizar una animación.

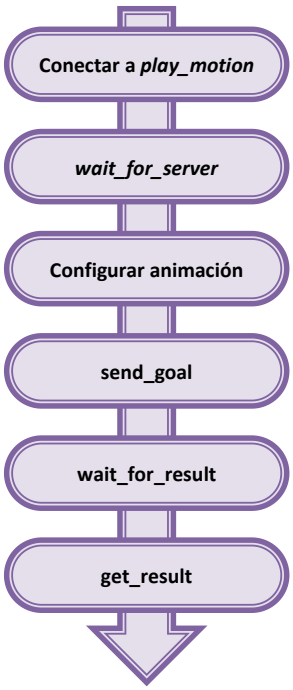


Figura 4.15: Flujo de acciones para la ejecución de una animación.

Como se puede observar el funcionamiento es similar al detallado en la Sección 4.2 para la ejecución de un movimiento simple. Se crea una conexión con el módulo *play-motion*, ROS permanece en espera hasta que reciba las indicaciones para realizar la animación deseada. Tras establecer la conexión se genera una meta, que en este caso

es el nombre de la animación y si se desea realizar de forma planificada. Si se realiza de forma planificada, el robot planifica el movimiento intentando evitar cualquier tipo de colisión, modificando para ello el movimiento en caso de ser necesario. Si no es planificada, el robot realizará las poses tal cual le han sido indicadas, con el riesgo de que se produzca alguna autocolisión. Si se diera ese caso, es posible que no se realizase. Los siguientes pasos, son los mismos que se siguen para establecer una postura en concreto. Se manda la meta *play-motion* y se queda a la espera del resultado de la animación ejecutada.

Para crear una animación, ésta está basada en una secuencia de posiciones ejecutadas en un tiempo determinado. Algunas animaciones están compuestas de una única posición, como es por ejemplo la animación predefinida *home*, con los brazos estirados junto al torso y cabeza hacia el frente, que directamente siempre acude a dicha posición independientemente del estado en el que se encuentre el robot. Para el resto de animaciones es necesario obtener una a una las posturas intermedias que se desean para reproducir cierto movimiento, indicando el instante en el que se debe enviar a ejecutar cada posición. Con esta secuencia, el robot irá pasando de una postura a otra, dando lugar al movimiento.

Para añadir las animaciones posibles que realizará el nuevo componente, se define el método *execute-animation(name)*

execute-animation(name):

Este método, es el utilizado para la ejecución de las animaciones. Recibe como parámetro el nombre de la animación. A continuación se indica el listado de las animaciones desarrolladas:

- **blinking:** simula un pestañeo, las figuras de la interfaz cambian del color blanco a negro durante un breve momento y vuelve a su color original de antes de comenzar la animación.

- **close-eyes:** acción que simula la acción de cerrar los ojos del robot NAO, dado

que para este trabajo se utilizan figuras, se les asigna a éstas el color negro con el tamaño completo de la figura para que se visualice correctamente.

- **dance-macarena:** animación que simula el baile de la Macarena y con la reproducción de la canción.
- **get-stronger:** esta animación del robot simula el gesto de estar poniéndose fuerte.
- **hello:** animación del robot que simula un saludo con la mano.
- **inhalation:** esta animación simula la acción de respirar, es decir, coger y soltar aire. El robot echa la cabeza hacia atrás y abre un poco los brazos, a la vez que se reproduce el sonido de inhalación de aire.
- **randomEyes:** es una animación en la que se cambia el color de las figuras para llamar la atención del paciente cuando se distrae. Realiza el cambio de color entre verde, amarillo, rojo y azul dos veces, estando el tamaño de las figuras al máximo.
- **rasta:** animación parecida a ramdomEyes, pero esta se reproduce más despacio y una única iteración del cambio de colores
- **Sit:** esta animación simula la acción de sentarse. Indicar que el robot REEM, no puede realizar esta animación por que su morfología no se lo permite al no disponer de piernas. Por lo que si se realiza una llamada a la misma únicamente se indica un mensaje que dice que el robot REEM no se puede sentar.
- **SitSleep:** en la arquitectura original, el robot se sienta y cierra los ojos. En este caso, para el nuevo robot REEM únicamente realiza la simulación de cerrar los ojos, que hace mostrando en la interfaz las figuras al máximo tamaño y de color negro.
- **SitWakeUp:** en la arquitectura original, el robot, abre los ojos y pestañea 2 veces. Para el nuevo componente, el robot REEM únicamente realiza la simulación

de abrir los ojos y pestañear, que en la interfaz se verán las figuras al máximo tamaño y de color blanco cambiando al color negro para simular el pestañeo.

- **Stand:** realiza la pose *home*, es decir, cabeza mirando al frente y brazos bajos, al lado del torso.

- **embrace:** el robot hace el gesto de estar explicando algo, es decir mueve los antebrazos hacia afuera y hacia adentro. Con este movimiento al realizar alguna explicación da la sensación de que el robot se expresa con gestos al igual que hace el ser humano.

- **searching:** en esta animación, el robot mueve la cabeza de un lado a otro como realizando una búsqueda del usuario. Esta animación puede ser utilizada por ejemplo, para simular que el robot no encuentra al usuario, por lo que le busca moviendo la cabeza a un lado y otro.

- **wipe_forehead:** esta animación permite simular la acción de limpiarse el sudor de la frente. Este gesto, es utilizado para la interactuar con el usuario indicándole que se encuentra cansado tras realizar la terapia.

En el Capítulo 5 del documento se muestra una demostración de la reproducción de estas animaciones.

Por último en relación a las animaciones, hay que indicar que para poder llevar a cabo la ejecución de las mismas es necesario cargarlas en el sistema. Es decir, ejecutar una sentencia que realice la carga del fichero de las animaciones predefinidas para este robot teniendo en cuenta que se deben cargar una vez se encuentre el *framework* ROS en ejecución. Comentar que esta carga se realiza junto con la serie de sentencias definidas en el script de ejecución de toda la arquitectura y está definido en el Apéndice A de este documento.

4.3.5. *Retargeting*

Una de las partes más importantes del componente, es la relacionada con el establecimiento de las poses de las articulaciones según lo indicado por el esqueleto de la Kinect, para que el usuario observe la postura que adopte el robot y la imite. Para la integración de las posiciones y el movimiento se han desarrollado los métodos *setAnglesFromVision* y *Retargeting*.

setAnglesFromVision(angles, mirrored):

Este método es el que se encarga de mandar al robot la instrucción de movimiento. Para ello, el método necesita conocer la lista de ángulos de las articulaciones partícipes de la postura y si el movimiento es espejado o no (información que recibe como parámetros). Si se le indica que la postura está espejada, el robot levantará el brazo contrario que el usuario, en caso contrario, moverá el brazo contrario ya que se encuentra enfrente del usuario. Esta cuestión es muy importante a la hora de realizar la terapia ya que si el usuario no es capaz de imitar la pose que toma el robot, este actúa en modo espejo adoptando la misma pose que el usuario para indicarle que esa es incorrecta, y a continuación le indica nuevamente cuál es la postura correcta. La lista de ángulos que se recibe viene definida del esqueleto de la Kinect, por lo que no se encuentra adaptada a las particularidades del robot en cuestión. Por este motivo es necesario realizar una función de *retargeting*, explicada a continuación, que adapte los ángulos recibidos a la morfología del robot. Conseguidos los ángulos transformados para el robot en cuestión, se realizan los pasos realizados en la Sección 4.2 para enviar la postura al robot y que es la realice.

***retargeting*(angles, mirrored):**

Es necesario indicar que el desarrollo de este método ha sido una de las partes más costosas de este desarrollo, dado los diversos inconvenientes encontrados y la dificultad de algunas de las trasformaciones realizadas.

Como ya se indicó en el Capítulo 2, cada robot tiene sus peculiaridades y por ello

no se pueden utilizar los ángulos que proporciona directamente la Kinect ya que no concuerdan con los que pudiera establecer el robot en sus propias articulaciones.

Inicialmente se pensó que dado que la transformación para el robot NAO ya estaba diseñada [Rossignoli 2015], para realizar la transformación de los nuevos robots de este proyecto no resultaría complicado realizar una pequeña modificación sobre esta para conseguir las nuevas funciones de transformación. Por ello, inicialmente se intentó realizar una transformación de las articulaciones haciendo pequeñas modificaciones en las funciones del robot NAO. Efectivamente este cambio fue satisfactorio para algunas de las articulaciones pero se encontró una gran dificultad en otras, especialmente en la articulación del giro del codo, ya que esta no seguía el mismo patrón de movimiento que el robot NAO porque en los nuevos robots depende del hombro. Como ya se ha indicado, además en esta articulación se observa una pequeña desviación respecto al torso en comparación con el NAO (ver Figura 4.16.

Para detallar en profundidad las transformaciones realizadas finalmente, se va a analizar en las siguientes secciones cada una de las articulaciones independientemente y la transformación que se ha realizado.

4.3.5.1. Apertura del hombro

La articulación de la apertura del hombro no depende de ninguna articulación más, como sí ocurre en otras articulaciones. Por tanto, para esta articulación ha utilizado el valor del ángulo facilitado por la Kinect aplicándole una reducción de precisión de menos de un 10º para solventar un pequeño desfase y conseguir mantener los brazos ligeramente mas abiertos, evitando colisiones en algunas posturas, como por ejemplo en la pose con los brazos hacia abajo.

Del mismo modo que el robot NAO, se utiliza exactamente el valor del ángulo de la Kinect tomando el seno del plano sagital (plano que divide el cuerpo en dos de forma perpendicular, ver Figura 2.16, Sección 2.4) aunque cabe destacar que para el brazo izquierdo se ha invertido el signo respecto al derecho, ya que los valores recibidos del

modulo visión están invertidos y para los REEM debe ser el mismo en ambos brazos.

Esta articulación asume el valor de 0º cuando el brazo se encuentra junto al cuerpo y 90º cuando el brazo se encuentra extendido, tal y como se muestra en el modelo original que se ha mencionado en el Capítulo 2.

4.3.5.2. Giro del hombro

El giro del hombro toma los mismos valores que el modelo del robot NAO, indicando el brazo hacia abajo con el valor 0º, el brazo extendido serían 90º y 180º cuando el brazo se encuentra hacia arriba. Para esta articulación se ha usado el plano transversal (ver Figura 2.16, Sección 2.4), división por la cintura. En el giro del hombro se usa el seno del ángulo recibido respecto al plano para los dos brazos, sin tener que realizar el cambio de signo. Al realizar esta transformación, se detectó que no todas las posturas eran correctas ya que esta articulación tenía una dependencia con la articulación de la apertura del hombro.

En la situación en la que el brazo se encuentra junto al cuerpo, el ángulo es 0º. En esta situación, se ha observado que la pose del robot respecto a la pose indicada por la Kinect difieren exactamente 90º y es por eso que el brazo se muestra incorrecto. Por otro lado, si la apertura del brazo toma el valor de 90º el brazo se muestra correctamente, por lo que no es suficiente con modificar en 90º el valor de la articulación.

Para dar solución a estas dos situaciones que deben tenerse en cuenta para la misma articulación, se ha desarrollado una fórmula (1) que proporciona más o menos peso a la implicación de la articulación de la apertura del hombro. La fórmula es la siguiente:

(1)

$$(1 - \frac{d}{\frac{\pi}{2}}) * (\frac{\pi}{2} - 0{,}10)$$

De tal forma que d es el valor de la articulación de apertura del hombro ya transformada para el robot. Esta corrección se añade al valor obtenido de la Kinect,

de modo que en el caso de que el brazo se encuentre junto al cuerpo (*d=0*) la fórmula tomará un valor próximo a 90º que al sumar al valor de la Kinect mostrará la postura correcta cuando el brazo se encuentre hacia abajo. Si el brazo esta abierto (*d=1*), el resultado de la fórmula dará un valor próximo a 0º y por ello la apertura del hombro no influirá en el ángulo del giro del hombro. Por tanto, gracias a esta fórmula, la postura será correcta en las dos situaciones, tanto cuando el brazo se encuentre cerrado como abierto. El valor 0,1 que se resta en el segundo multiplicador, se ha comprobado que es el valor medio de error para que la postura tomada sea correcta. En consecuencia las poses realizadas son más semejantes a la realidad.

Por último se ha incluido una pequeña modificación sobre todo el valor obtenido multiplicando este por 1,3 para realizar una ajuste sobre la postura, dado que al situar el brazo en 90º visualmente no quedaba con ese ángulo, es decir, no quedaba recto, se mostraba ligeramente hacia abajo. Esta última modificación es consecuencia de la propia arquitectura del robot, ya que se ha observado que el hombro no forma un ángulo exacto de 90º respecto al torso, si no que tiene un pequeña desviación como ya se comentó en la Sección 2.3 y se puede ver en la Figura 2.10 de esa sección.

4.3.5.3. Apertura del codo

El ángulo de la apertura del codo coincide con el ángulo obtenido con el modelo original del NAO, tomando valor 0º si el codo se encuentra extendido y el valor 90º con el codo flexionado. Esta articulación es totalmente independiente, no depende de ninguna otra. Al igual que el robot NAO, se utiliza el coseno del plano sobre el ángulo de la Kinect [Rossignoli 2015], pero para los nuevos robots se invierte el valor porque la diferencia de los planos con los nuevos robots hacia que se visualizará el antebrazo en sentido inverso. Adicionalmente, ha sido necesario restar 180º al valor del coseno para poder visualizar correctamente los ángulos recibidos esperados para esta articulación. En la Figura 4.16 se muestra el robot con el brazo izquierdo (a la derecha de la imagen) sin modificaciones y el brazo derecho con la postura correcta.

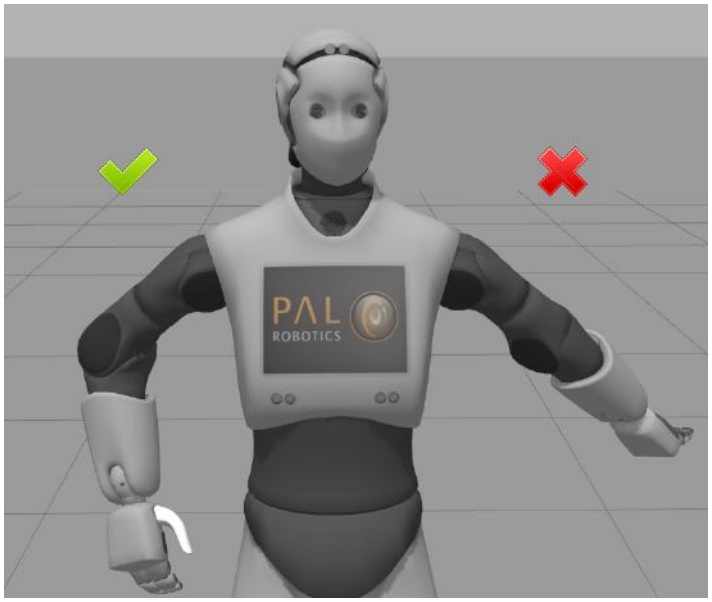


Figura 4.16: Transformación de la apertura del codo. Brazo derecho correcto (con transformación), brazo izquierdo incorrecto (sin ella)

4.3.5.4. Giro del codo

La articulación del giro del codo ha sido la mÁs complicada de desarrollar y requiere diversas transformaciones para llegar a poder obtener valores válidos. En el modelo inicial ya se vio una dependencia con la posición del hombro y para realizar la transformación se implementó la fórmula descrita en la Sección 2.4.2 de este mismo documento. En el robot NAO, usando dicha fórmula es posible obtener una adaptación casi perfecta de la articulación del giro del codo, pero en el robot REEM y REEM-C dicha fórmula no es suficiente por la morfología propia de estos robots y las dependencias con distintas articulaciones. Por este motivo, se han tenido que llevar a cabo varias transformaciones para realizar el ajuste del valor de este ángulo. Estas conversiones se han hallado de forma empírica siguiendo un proceso de prueba y error. Inicialmente se intentaron calcular todas las variables posibles mediante combinatoria, es decir combinaciones de valores con ajustes similares a los de las demás articulaciones. Se implementó un pequeño programa en Java que realizaba el calculo de todas las po-

sibles combinaciones simples similares a las anteriores ya aplicadas, pero el resultado obtenido con esas fórmulas nunca era superior al 50 % de las poses de forma correcta por lo que se descartó esta idea inicial.

Por ello, teniendo presente la fórmula inicial del modelo tomado de la función de transformación del NAO (1):

(1)

$$\frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}})$$

donde:

- a = Ángulo de giro del codo calculado mediante el vector que une el hombro a la cadera.
- b = Ángulo de giro del codo calculado mediante el vector que une los hombros.
- c = Ángulo de apertura del hombro.

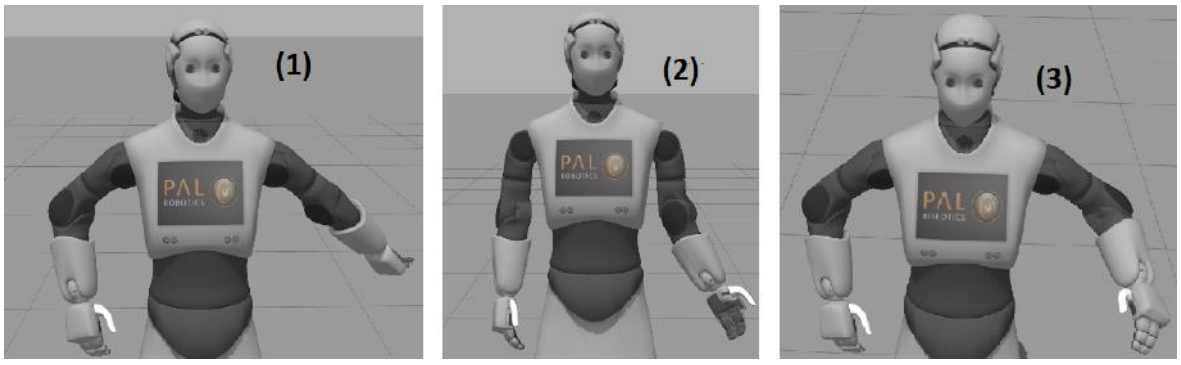


Figura 4.17: (1) Primera transformación (2) Segunda transformación (3) Tercera transformación.

Se han realizado distintos análisis y añadidos a esta fórmula para obtener el mayor número de poses correctas. Primeramente, a la fórmula inicial se le ha añadido la misma fórmula que se ha tenido en cuenta en la articulación del giro del hombro, porque el giro

del codo también es dependiente de la articulación de giro del hombro. En la primera imagen (1) de la Figura 4.17 se muestra como el brazo izquierdo del robot (derecha en la imagen) no se le ha aplicado esta primera transformación mientras que en el brazo derecho sí la tiene en cuenta, mostrándose este último con la pose correcta. La fórmula exactamente utilizada es (2):

(2)

$$(1 - \frac{d}{\frac{\pi}{2}})\frac{\pi}{2}$$

Gracias a esta fórmula añadida se solucionan los problemas siempre que el brazo se encuentra extendido, es decir, cuando la apertura del codo sea 0º. En este caso, cuando la apertura del hombro sea 0º, con la fórmula (1) se obtendrá un valor cercano a 90º y se aplicará esta corrección en el giro del codo.

Pero existe otro inconveniente adicional en el caso de que el brazo se encuentre extendido ($d = 90^o$). Visualizando la segunda imagen (2) de la Figura 4.17, cuando la apertura del hombro y la del codo tiene un valor de 90º, se produce un desfase de 90º (brazo derecho posición correcta con la transformación, brazo izquierdo erróneo). Para esta situación se ha desarrollado la siguiente fórmula añadida a la fórmula ya conseguida (3):

(3)

$$\frac{d}{\frac{\pi}{2}} * (\frac{e}{\frac{\pi}{2}} * \frac{\pi}{2}) = \frac{2ed}{\pi}$$

donde:

- **d:** Apertura del hombro
- **e:** Apertura del codo.

Gracias a este añadido a la fórmula, si el brazo se encuentra extendido ($d = 90^o$) y el codo flexionado ($e = 90^o$) se incluye una modificación de 90º que realiza la corrección de la postura que se puede visualizar en la imagen (3) de la Figura 4.17. En concreto

esta fórmula añadida se resta al resto de la fórmula ya que en caso de sumarse se ampliaría el desfase inicial a 180º.

Con estas transformaciones, las posturas eran correctas pero de forma inversa y por ello se ha realizado un cambio de signo al conjunto de la fórmula de transformación dando lugar a la siguiente fórmula final (4):

(4)

$$RotC = \frac{ac}{\frac{\pi}{2}} + b(1 - \frac{c}{\frac{\pi}{2}}) - \frac{\pi}{2} + d + \frac{de}{\frac{\pi}{2}}$$

Para realizar la explicación, facilitar la comprensión de las transformaciones realizadas y dado que fue la forma de obtener la expresión final, se ha utilizado la fórmula expandida pero para la versión final del componente se ha realizado una simplificación de la misma, dando lugar a la siguiente formula (5):

(5)

$$b + d + \frac{4(ac - bc + de) - \pi^2}{2\pi}$$

Como se observa, la expresión resultante para la transformación del ángulo de esta articulación es bastante más compleja de lo esperado inicialmente, por lo que hace pensar que la elección de los planos sobre los que se ha elaborado la función no hayan sido la mejor elección. Independientemente de los planos elegidos es visible que la arquitectura del robot causa que una función de *retargeting* sea bastante complicada. Una diferencia con el robot original NAO, como es la desviación del hombro respecto al torso, puede generar y genera como se ha visto en este desarrollo, muchísimos problemas para la realización de la función de *retargeting*. En la sección de posibles trabajos futuros se destacará la posibilidad del uso de otros planos para simplificar y mejorar el *retargeting*.

4.3.5.5. Giro de la muñeca

Por último, a pesar de que la Kinect no es capaz de reconocer de forma totalmente correcta pero dado que en las poses establecidas así se requería, se ha realizado la conversión de la articulación del giro de la muñeca. Gracias a esta articulación se podrá mover en algunas posturas la palma para orientarla para el lado deseado.

Para esta articulación la modificación realizada ha sido mínima ya que la transformación que se estaba utilizando en el modelo original era válida para el nuevo robot. La modificación mínima realizada es que se ha invertido el signo para una de las muñecas ya que al realizar la comprobación se veía como siempre se mostraba invertido. Como ya se indicó en el Capítulo 2 esta articulación abarca desde los 88º a los -88º dependiendo del resto del brazo para realizar la pose completa, es decir, no se puede indicar con qué ángulo mira hacia adentro o fuera ya que esta posición varia según la postura del resto del brazo.

Con estas transformaciones de la función de *retargeting* es posible realizar una gran parte de las 34 posturas requeridas inicialmente. En el Capítulo 5 se expondrá detenidamente los resultado correctos obtenidos gracias a las transformaciones.

4.4. Implementación de ReemCompC

Una vez se consiguió un correcto funcionamiento del componente ReemComp se decidió realizar un añadido en el que se incluyera este nuevo componente a la arquitectura. Para realizar esta integración, partiendo del componente ReemComp se ha realizado un análisis con las similitudes con el componente y las posibilidades de integración de ReemCompC con el robot REEM-C.

4.4.1. Métodos de control

Dado que el robot REEM-C utiliza las mismas herramientas para su funcionamiento que el robot REEM y estas son el *framework* ROS y el simulador Gazebo, los métodos de control desarrollados para el componente ReemComp son totalmente reutilizables para el componente ReemCompC. Por lo tanto se ha incluido la implementación de las funciones *isSimulated()* y *isConnected()* desarrolladas inicialmente para el robot REEM en el componente ReemCompC. Añadiendo estas funciones, se tiene en cuenta la misma funcionalidad que se tiene en el componente ReemComp.

4.4.2. Reproducción de audio y Text-to-Speech

Para incluir las funciones realacionadas con la interacción hablada del robot, igual que con los métodos de control se ha realizado un estudio para verificar el modo de incluir esta funcionalidad para el robot REEM-C. Tras este estudio, se ha concluido, que el robot REEM-C tiene las mismas caracteristicas para realizar esta función que el robot REEM. Por lo tanto, se han reutilizado las funciones *playAudioFile(file,blocking)* y *say(speech,blocking)* para incluir la funcionalidad requerida según la arquitectura original.

4.4.3. Interfaz

Respecto a la interfaz, se ha diseñado una interfaz final de tal forma que cualquier sistema operativo basado en UNIX pueda mostrar y utilizar la misma ya que el robot REEM, como ya se ha indicado, dispone de un sistema operativo basado en la interfaz Openbox. Esta caracteristica facilita la posibilidad de incluir la interfaz en el robot REEM-C pero si se obseva su arquitectura, dicho robot no dispone de ninguna pantalla para mostrar información multimedia.

Esta cuestión supone un inconveniente, ya que inicialmente se pensó que el robot

REEM-C no podría mostrar la interfaz diseñada. Por este motivo, se retomó la idea de los leds, al igual que se utilizan en el robot NAO. Se estudió la arquitectura completa del robot y se vio que el robot REEM-C únicamente dispone de leds en la boca. Esta es una parte demasiado pequeño para mostrar un *feedback*, además de ser un elemento único y para la arquitectura NAOTherapist se requieren dos componentes (uno para cada brazo).

Adicionalmente en el análisis realizado, se observo que este robot dispone de conexión wifi y dos ordenadores internos que pueden facilitar la conexión con una pantalla o dispositivo externo. Siendo esto así, se ha decidido usar la misma interfaz diseñada para el robot REEM, pero en este caso, en una simulación real, se visualizará en un componente externo que podrá ser conectado al robot por una conexión como por ejemplo HDMI.

4.4.4. Animaciones

En relación a las animaciones, como ya se había observado en la Sección 4.2 para establecer una postura los pasos a seguir son los mismos para ambos robots, por lo que para realizar el diseño de una animación en el robot REEM-C se podía seguir la misma secuencia de acciones que en el componente ReemComp.

Siendo esto así, se han incluido en el componente ReemComp todas las animaciones diseñadas para el componente ReemComp añadiendo algunas animaciones que por la morfología del robot REEM-C si son posibles en del mismo modo que en el robot NAO, como por ejemplo la animación de sentarse. Por lo tanto, adicionalmente a las animaciones ya definidas para el componente ReemComp, se han incluido para el componente del REEM-C las siguientes animaciones:

- **Sit:** el robot REEM-C realiza la acción de sentarse, dentro de sus posibilidades, que es posicionándose en cuclillas. Este robot es mucho mas pesado que el NAO por lo que realizar la animación al completo sentándose en el suelo sería muy

arriesgada facilitando la caída del robot y la dificultad de volver a levantarse.

- **SitSleep:** en la arquitectura original, el robot se sienta y cierra los ojos. Por ello en esta animación el robot REEM-C, se sienta de la forma que se ha indicado para la animación anterior y muestra en la interfaz las figuras en negro para simular el cierre de ojos.

- **SitWakeUp:** en la arquitectura original, el robot se levanta, abre los ojos y pestañea 2 veces. Por lo tanto el robot REEM-C, se sienta y realiza la simulación de los ojos del mismo modo que el robot REEM.

- **Stand:** realiza la pose *home*, es decir, cabeza mirando al frente y brazos bajos, al lado del torso. Además si el robot se encontraba sentado, se levanta.

Para realizar la ejecución de las animaciones en el componente ReemCompC se hace uso de la función **execute-animation(name)** semejante a la del componente ReemComp. Del mismo modo que para el robot REEM, es necesario cargar las animaciones para su ejecución. Igualmente esta carga se realiza junto con la serie de sentencias definidas en el script que ejecuta el sistema y que se describe en el Apéndice A de este documento.

4.4.5. *Retargeting*

Inicialmente se pensó que dado que el robot REEM-C es un modelo posterior al robot REEM, era posible que no se tuviera el inconveniente del desvio de la articulación del hombro. Tras realizar el análisis del mismo, detallado en el Capítulo 2, se observó que ambos robots tenían la misma arquitectura en la parte superior, es decir, tienen las mismas articulaciones y capacidad de movimiento en las extremidades superiores, por lo que se seguía teniendo el inconveniente del desvio del hombro y con ello los problemas relacionados con la articulación de giro del codo.

A pesar de seguir teniendo este inconveniente, dado que la morfología de la parte superior del robot REEM-C es igual a la del robot REEM, es posible utilizar la misma función de conversión de angulos que se utiliza en el componente ReemComp. Por ello, se ha incluido en el componente ReemCompC las mismas funciones que en el componente ReemComp para realizar el movimiento.

Es necesario indicar, que dado que el robot REEM-C tiene piernas, es bastante mas inestable que el robot REEM, por lo que es sencillo que este robot pueda caerse. Dado que para este proyecto únicamente se utiliza un simulador, no es posible comprobar si con el *retargeting* implementado el robot podría llegar a caerse al realizar alguna postura.

4.5. Integración terapias

Integrados los dos nuevos componentes en la arquitectura, es el momento de utilizarlos para el dominio elegido para este proyecto, que son las terapias. Este dominio se encuentra incluido en la arquitectura original, en concreto en el componente *PELEA*. El domino, se compone por una serie de acciones que son ejecutadas en función de varios parámetros y precondiciones dando lugar al resultado deseado. Después el componente ejecutivo coge cada una de esas acciones de "nivel medio" que envía PELEA y las descompone en instrucciones de bajo nivel, que son las que va recibiendo el componente Robot.

Dado que el dominio PDDL ya se encontraba diseñado, no se ha analizado en profundidad el funcionamiento del mismo ya que dicho análisis se encuentra incluido en el origen del proyecto NAOTherapist. Sí es cierto que para la realización de este trabajo ha sido necesario conocer la funcionalidad del mismo para poder crear los problemas específicos para los nuevos componentes.

En los nuevos problemas creados, se ha elegido una muestra del total de poses disponibles. Con esta recopilación de poses, se indican en el problema las poses a

realizar y su duración para cada sesión. Una terapia tiene S sesiones, cada sesión tiene E ejercicios que a su vez disponen P poses de una duración Di determinada. En el Capítulo 5 de este mismo documento, en la evaluación del sistema, se han aplicado los problemas definidos.

Es necesario indicar que para la realización de la terapia, las poses realizadas se visualizan de forma invertida, de tal forma que el paciente que realice la terapia pueda imitar al robot como si fuera un espejo. Esto es un punto importante ya que facilita al usuario la comprensión de la pose que debe ejecutar.

Capítulo 5

Evaluación del sistema

En este capítulo se describe la evaluación del sistema desarrollado, donde se detalla la experimentación realizada que abarca toda la funcionalidad desarrollada en este proyecto. Además, gracias a esta experimentación se puede demostrar que los objetivos planteados en el primer capítulo del documento han sido cumplidos.

Los vídeos de la experimentación de este proyecto se han subido al canal oficial del proyecto NAOTherapist, ya que tras la finalización de este proyecto los nuevos componentes han sido incluidos dentro de la arquitectura. Por ello, todos los vídeos mencionados en este capítulo se pueden encontrar en dicho canal en una lista de reproducción especifica.

https://www.youtube.com/playlist?list=PL1Fs33N0K0mZseYF4qz-MLZH0-yyP37o-

5.1. Evaluación del *retargeting*

En esta sección se detallan los resultados de la evaluación de la función de *retargeting* implementada. Para llevar a cabo la evaluación de se ha utilizado la herramienta **RetargetingHelper** (ver Figura 5.1) que ofrece una lista con las 34 posturas incluidas en la arquitectura NAOTherapist mostrando los valores que toma cada articulación del

esqueleto de la Kinect. Al enviar la pose indicada, se ejecuta la función de *retargeting* del robot para que tome dicha postura.

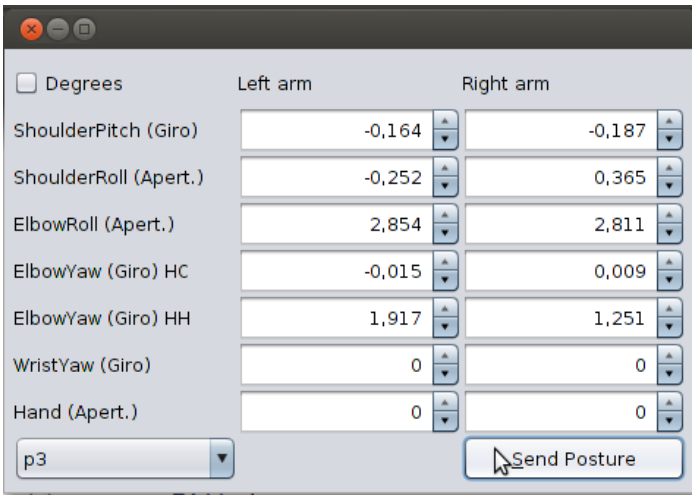


Figura 5.1: Herramienta RetargetingHelper.

Para este trabajo se ha modificado el fichero de ejecución de la herramienta para poder visualizar simultáneamente el robot NAO original de la arquitectura en su simulador Choregraphe y el robot REEM o REEM-C con Gazebo.

Dado que se ha comprobado que la función de *retargeting* es válida, en modo simulación, para los dos nuevos componentes desarrollados en este trabajo, para realizar la evaluación de esta función se utilizará el robot REEM-C asumiendo que los resultados serían exactamente los mismos para el robot REEM. Se han elegido 8 poses representativas de las 34 disponibles y se reproducen en el robot NAO y REEM-C.

Gracias a esta herramienta se pueden observar las diferencias entre la pose tomada por cada robot para cada una de las poses predefinidas en NAOTherapist. Es necesario indicar que esta diferencia es meramente visual ya que se pretende que ambos robots tomen la pose lo más similar posible pero por la morfología de cada uno de ellos es imposible medir la similitud de forma exacta.

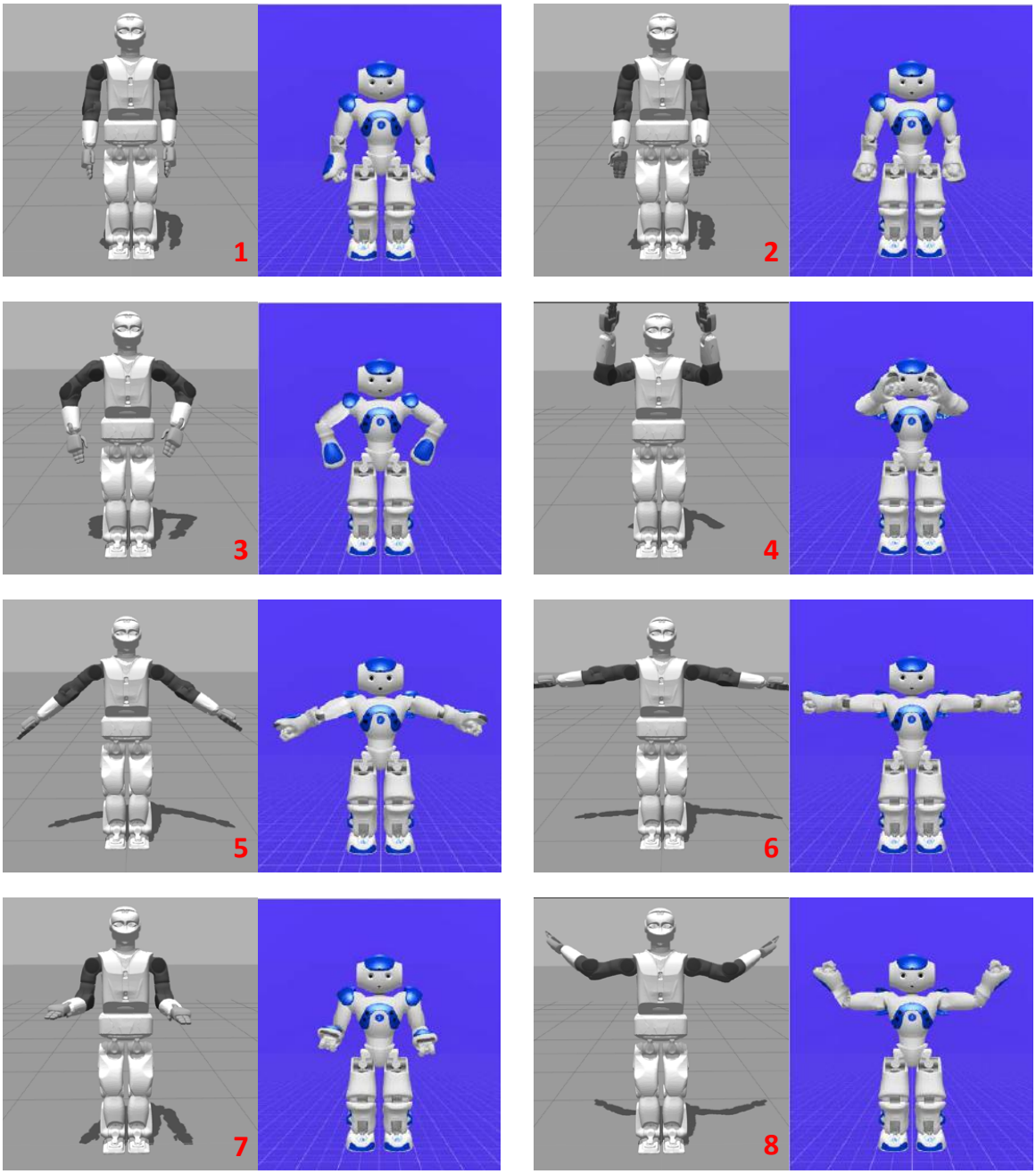


Figura 5.2: Comparativa de poses entre el robot REEM-C y NAO usando función retargeting.

En la Figura 5.2 se muestra el resultado de la evaluación de la que se extraen diversas conclusiones. En la primera imagen (1) de la figura se muestra la pose inicial (*home*) que como se ve se ha conseguido reproducir exactamente igual que en el robot NAO. Con la segunda imagen (2) se muestra que la articulación del giro de la muñeca funciona correctamente ya que el nuevo robot orienta la muñeca del mismo modo que el robot NAO.

En la imagen (3) se muestra una pose que simula que el robot tiene los brazos en la cintura (comúnmente pose *en jarras*) y se observa como el nuevo robot sitúa las manos ligeramente por debajo de la cintura. Esto es debido a que las poses predefinidas han sido tomadas para el robot NAO, que como se puede observar claramente también en las imágenes (5) y (6) tiene los brazos bastante más cortos que el nuevo robot. A pesar de ello la postura se visualiza correctamente y el nuevo robot es capaz de flexionar el antebrazo y orientarlo correctamente hacia las caderas.

En la cuarta imagen se visualiza como el robot NAO coloca las manos en frente de la cabeza pero el nuevo robot Es debido al tamaño de la cabeza del robot NAO, por lo que al convertir la postura al nuevo robot éste posiciona las manos tomando como referencia la cabeza más ancha del NAO. También influye el hecho ya comentado de la desviación del hombro respecto tronco del nuevo robot. Por este mismo motivo, en la imagen (7) se visualiza cómo el nuevo robot toma correctamente la postura general aunque con los antebrazos ligeramente orientados hacia afuera (en vez de adelante como el robot NAO). Esto mismo sucede con las posturas de la imagen (9) y (10)

Por último, en la imagen (8) se muestra que no solo influye la morfología del robot. Incluso después de realizar diversas trasformaciones para el ángulo del codo, la función de *retargeting* podría mejorarse para este tipo de poses en las que tanto el ángulo de apertura del codo y hombro son cercanas a 90^{o} ya que se muestra una ligera desviación de la postura correcta. Dependiendo del ángulo exacto que se tome en estas dos articulaciones si visualizará una mayor o menos desviación.

En relación al tiempo de respuesta y velocidad de movimiento se ha observado que el robot NAO es algo más rápido que los nuevos robots e incluso el robot REEM se mueve más ligero que el robot REEM-C. Con las herramientas disponibles actualmente no es posible facilitar una mayor velocidad a los nuevos robots.

Finalmente se puede concluir que la nueva función de *retargeting* desarrollada funciona correctamente a pesar de que es posible llevar a cabo una mejora de la misma y que por tanto queda cumplido el objetivo de la integración del movimiento del robot en la arquitectura NAOTherapist.

5.2. Evaluación de la interfaz gráfica

En esta sección se detalla la experimentación realizada en relación con la interfaz gráfica encargada de proporcionar el feedback al usuario.

5.2.1. Corrección de la postura

Tal y como se ha explicado en el capítulo de la implementación, el robot original NAO utiliza los leds de los ojos para proporcionar un *feedback* al usuario e indicarle de forma gradiente si la pose indicada es correcta. Dado que el robot REEM y REEM-C carecen de algún tipo de leds lo suficiente visibles que pueda facilitar este *feedback* se ha desarrollado la interfaz que muestra con figuras que aumentan de tamaño cuando de correcta es la pose que se esta realizando. Esta información es independiente para cada brazo, de tal forma que si se tiene un brazo mal colocado y el otro bien, así se representará en la interfaz distinguiendo el brazo izquierdo (figura del lado izquierdo) del brazo derecho (figura del lado derecho). Si la pose es totalmente correcta se mostrará la figura al máximo tamaño y si es totalmente incorrecta se mostrará al mínimo tamaño.

Para realizar la prueba de esta parte del componente, conviene indicar que la interfaz es común para ambos componentes ya que no ha sido necesario realizar ninguna

modificación adicional para adaptar la interfaz diseñada para el componente del REEM en el componente del REEM-C por lo que la evaluación mostrada es idéntica para ambos robots. Para demostrar el correcto funcionamiento de la interfaz se han tomado 2 poses diferentes. El sistema revisará si la postura del usuario que se encuentra en frente de la Kinect es igual que la pose del robot, mostrando en la interfaz lo que falta para llegar a la postura correcta. Para cada una de las posturas se muestra una serie de experimentos unitarios con el que se irá demostrando el correcto funcionamiento de la interfaz para cada posible casuística.

5.2.1.1. Pose A

Este primer experimento se realiza con el robot REEM y con la interfaz morada con círculos como figuras del feedback. La pose de prueba utilizada es una en la que ambos brazos se encuentran en paralelo al suelo, es decir, como si fuera un avión. En la Figura 5.3 se puede observar el resultado de esta primera evaluación, en la que se enseña la pose del robot (derecha y siempre la misma), el esqueleto de la Kinect (varía según se mueva el usuario) y el feedback que proporciona la interfaz que depende de la diferencia entre la postura correcta y la que realiza el usuario. En esta figura, se puede observar en la primera fila (1) que la postura que toma el usuario es exactamente igual que la que se visualiza en el robot, por lo que la interfaz (a la izquierda) toma la postura como exacta mostrando la figura central del círculo al máximo tamaño para ambos lados (derecha e izquierda).

Sin embargo, en la segunda imagen de la Figura 5.3 se ve como la pose que está tomando el usuario (esqueleto de la Kinect) es totalmente incorrecta y por ello no se visualiza nada del círculo central en la interfaz, únicamente se puede apreciar un pequeño punto que indica que se está reconociendo la articulación.

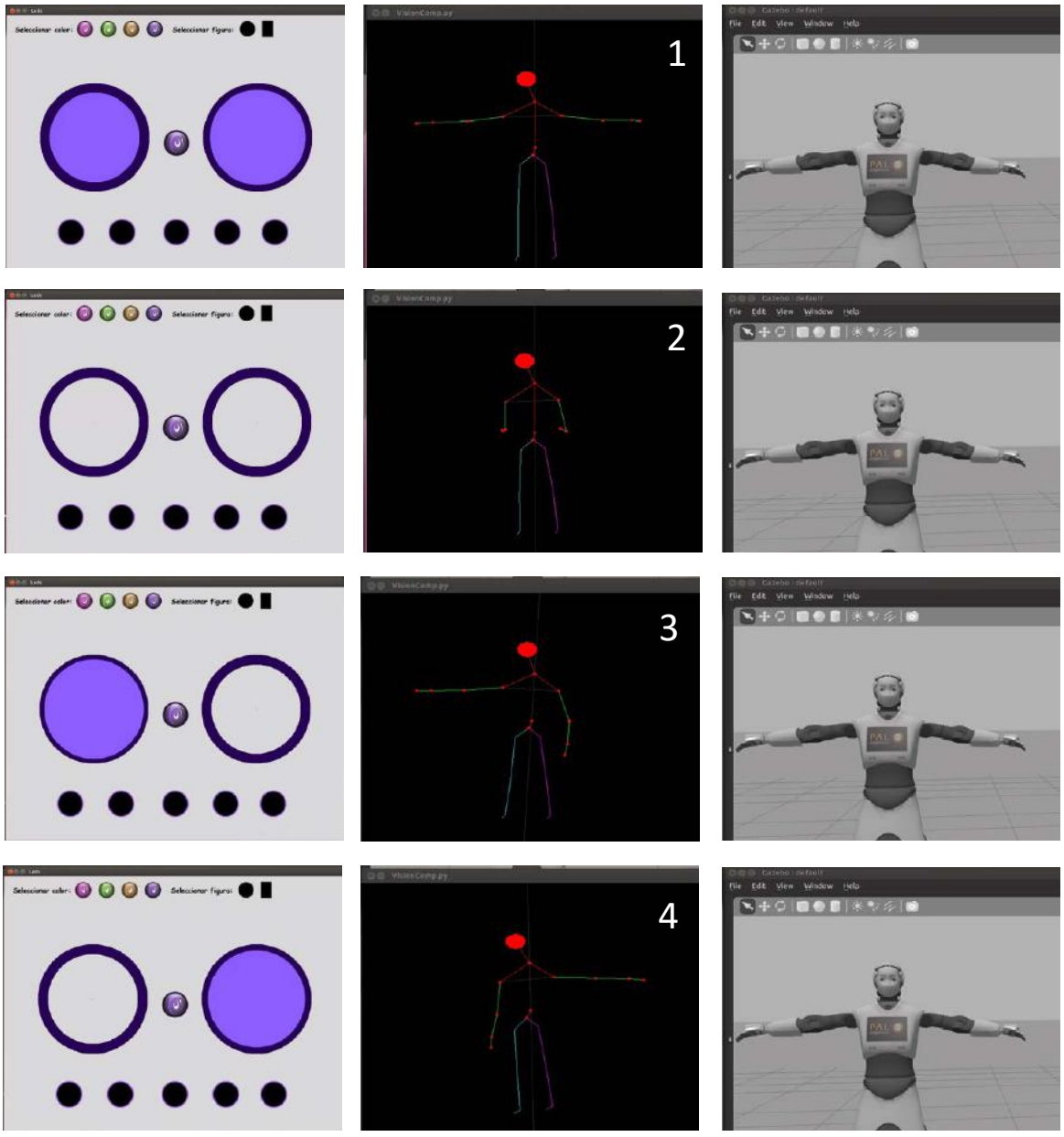


Figura 5.3: Evaluación de la pose A con interfaz con figura circular, esqueleto de la Kinect y robot REEM .

Por último, se observa en la tercera (3) y cuarta (4) fila de la figura que es uno de los brazos el que está correcto y el otro incorrecto. Así en la tercera fila se observa como el brazo izquierdo está perfectamente situado pero el izquierdo se encuentra totalmente incorrecto. Sucede justo al contrario en la siguiente fila, en la imagen (4).

Gracias a este experimento queda por tanto demostrado el correcto funcionamiento de la interfaz con el robot REEM cuando la figura que proporciona el feedback es un círculo, aunque se debe indicar que si el robot fuera el robot REEM-C el funcionamiento sería el mismo ya que la interfaz es válida para ambos robots.

5.2.1.2. Pose B

El segundo experimento se realiza con el robot REEM-C y con la interfaz morada pero con barras como figuras del *feedback*. La pose de prueba utilizada es una en la que el brazo de la izquierda esta hacia arriba y el brazo de la derecha ligeramente flexionado, como "en jarras". En la Figura 5.4 se observa al igual que en el ejemplo anterior, que en la primera fila (1) se visualiza al usuario correctamente posicionado por lo que el *feedback* de la interfaz es totalmente correcto mostrando las barras en su máximo tamaño.

En relación con esta pose, se muestra en la segunda fila de la figura cómo la postura que está tomando el usuario se aproxima a la postura correcta, aunque no llega a ser la postura deseada. Por lo que se ve en el esqueleto de la Kinect, el usuario debe subir un poco más el brazo de la izquierda y flexionar el brazo de la derecha.

Por último, en la tercera fila se muestra como el usuario ha realizado la postura completamente inversa a la que realiza el robot, por lo que la interfaz le indica que la postura es incorrecta y no muestra nada en las figuras de feedback.

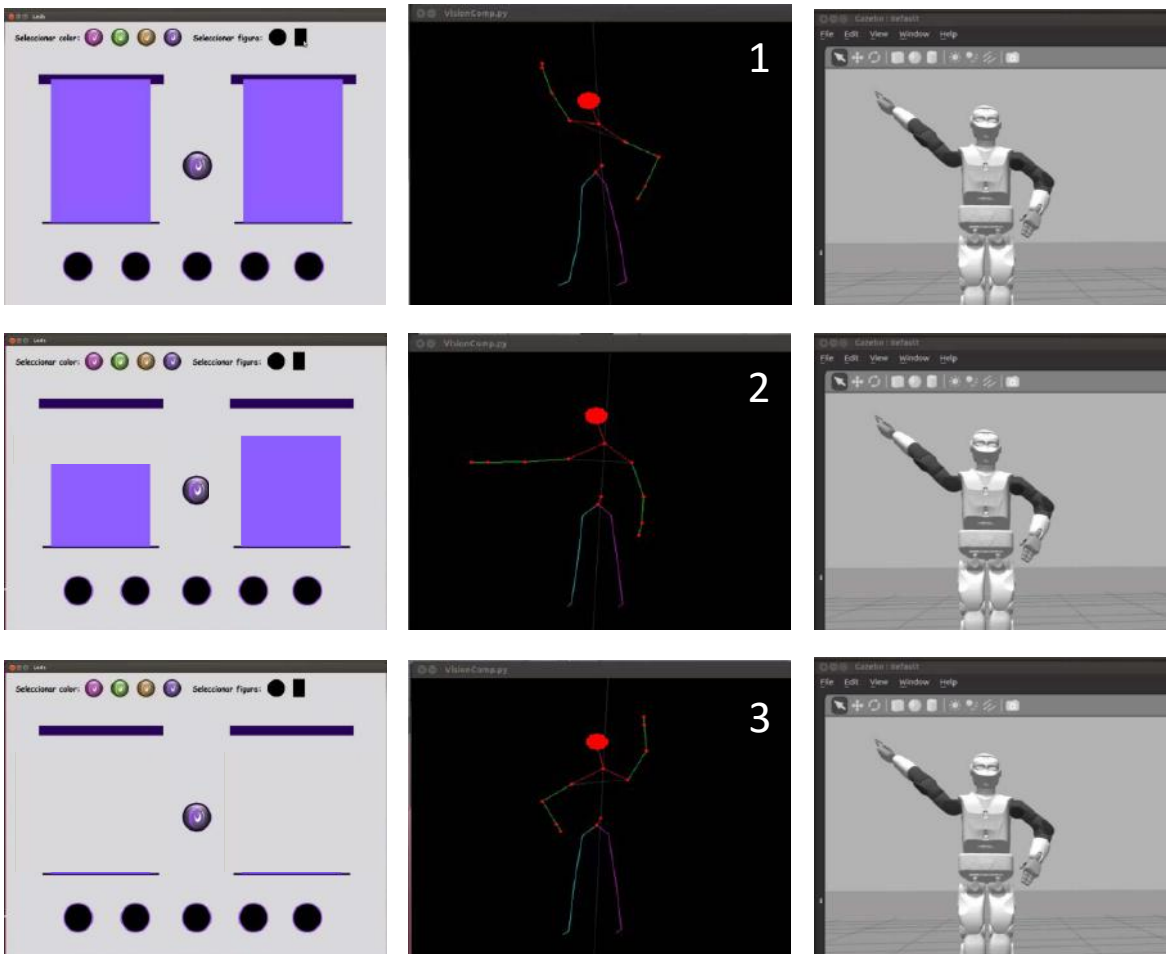


Figura 5.4: Evaluación de la pose A con interfaz con figura rectangular, esqueleto de la Kinect y robot REEM-C

Gracias a este experimento queda por tanto demostrado el correcto funcionamiento de la interfaz en el componente del robot REEM-C cuando la figura que proporciona el feedback es una barra/rectángulo. Si el robot fuera el robot REEM el funcionamiento sería el mismo ya que la interfaz es válida para ambos robots.

5.2.2. Botones

Para la experimentación de los botones incluidos en la interfaz que simulan a los botones físicos del robot NAO y demostrar la correcta conexión con la arquitectura, se ha realizado una validación con el botón *FrontTactil*, que es el primer botón empezando por la izquierda de la interfaz y es uno de los que actualmente tienen funcionalidad. Este botón al ser pulsado pausa la ejecución del sistema y al pulsarlo nuevamente reanuda la ejecución.

Dado que el desarrollo de la interfaz es común para ambos componentes, esta validación únicamente se ha realizado para uno de ellos ya que en el otro el funcionamiento será igualmente correcto.

A continuación se muestra la traza que facilita el componente desarrollado y el componente Ejecutivo (Executive) de la arquitectura durante la ejecución de la terapia, de modo que se visualiza el momento en el que se pausa y reanuda la ejecución al volver a pulsar.

```
Last button pressed set to FrontTactil
->getLastButtonPressed: frontTactil
```

Así se informa al Ejecutivo que el botón fue pulsado y es el Ejecutivo quien realiza la acción de pausar y reanudar la ejecución del sistema. A continuación se muestra la traza correspondiente al componente Ejecutivo:

```
1 DETECT-PATIENT [‘reem’, ‘pt00’]
2 IDENTIFY-PATIENT [‘reem’, ‘pt00’]
3 GREET-PATIENT [‘reem’, ‘pt00’]
4 START-TRAINING [‘reem’, ‘loc_training’, ‘pt00’, ‘loc_show’, ‘ses10’]
Emergency: 0
5 PAUSE-SESSION [‘reem’, ‘pt0’, ‘ses1’]
6 RESUME-SESSION [‘reem’, ‘pt0’, ‘ses1’]
7 INTRODUCE-EXERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘ses1’]
8 START-EXEERCISE [‘e100r’, ‘nao’, ‘pt00’, ‘stand_up’]
9 EXECUTE-POSE [‘p_id0’, ‘e100r’, ‘p6’, ‘t1’, ‘nao’, ‘pt00’, ‘stand_up’]
```

La ejecución se descompone en acciones de bajo nivel (programadas en los nuevos componentes). El botón se pulsa durante la acción *START-TRAINING* y seguidamente se pausa la ejecución. Al pulsar nuevamente el botón (*RESUME-SESSION*) se reanuda la ejecución del sistema para continuar con la pose correspondiente.

Con esta experimentación queda demostrado el correcto funcionamiento de la comunicación entre el robot y la interfaz además de la posibilidad de añadir multitud de funcionalidades a estos botones a través del componente Ejecutivo.

5.2.2.1. Elección de skin

Para llevar a cabo la experimentación relacionada con el cambio del modelo de la interfaz (skin) se ha utilizado un script de pruebas que inicializa todo el sistema y permite ejecutar cada una de las funciones del componente por separado. Este fichero de prueba ha sido de mucha utilidad a la hora de diseñar la interfaz ya que permite visualizar la misma e indicar las funciones que se desean ejecutar sin tener que depender del componente Ejecutivo de la arquitectura de NAOTherapist.

Esta prueba se ha realizado una única vez para los dos componentes dado que como ya se ha indicado a lo largo del documento la interfaz diseñada es válida para los dos robots.

Por tanto, para demostrar el correcto funcionamiento de los botones que corresponden a la funcionalidad del cambio de skin, se ha utilizado el fichero de prueba que ejecuta en todo momento el método *getLastButtonPressed* de tal forma que este continuamente esperando a saber si se ha pulsado un botón. Para ver mas claramente la funcionalidad se ha grabado un pequeño vídeo denominado **Elección de skin - Evaluación REEM / REEM-C** en el que se muestra el cambio de todas las skins tanto colores como formas siempre.

La traza ejecutada por esta funcionalidad es similar a la que se indica en la prueba de los demás botones pero teniendo en cuenta que el Ejecutivo no muestra traza del cambio de interfaz ya que esta funcionalidad es única en los dos nuevos componentes creados. Siendo esto así, en la traza del componente se visualizará por ejemplo al querer cambiar a la figura del circulo la siguiente traza:

```
Last button pressed set to middleTactil
->getLastButtonPressed: circulo
->updateSkin: circulo
```

Para concluir, con la experimentación realizada en esta sección queda demostrado el correcto funcionamiento de la interfaz, los botones de la misma y el cambio del modelo de la interfaz de modo que se le podrá facilitar al usuario el feedback de la pose tomada y además incentivarle con la posibilidad del cambio de la visualización de la pantalla.

5.3. Animaciones y reproducción de audio

Para realizar las pruebas correspondientes a las animaciones y la reproducción del audio se ha creado una secuencia de ejecución con todas las animaciones desarrolladas para mostrar el correcto funcionamiento de todas ellas tanto para el robot REEM como para el robot REEM-C. El resultado de esta secuencia de ejecución de las animaciones se muestra en el vídeo **Animaciones - Evaluación REEM / REEM-C** que se encuentra disponible en el canal de Youtube mencionado al comienzo del capítulo.

En ambos vídeos se puede observar que para algunas animaciones como por ejemplo la animación *hello* o *get-stronger* se ha probado junto con la reproducción de ficheros de audio de tal modo que gracias a este experimento queda demostrado el correcto funcionamiento de las animaciones, la reproducción de ficheros de audio y la correcta fusión de ambas funcionalidades. Cabe destacar que la reproducción de los ficheros de audio es no bloqueante, de modo que ambas funcionalidades comienzan casi en el mismo instante.

En otras animaciones como por ejemplo *rasta* y *blinking* únicamente son realizadas por la interfaz mientras que el robot queda a la espera de la siguiente acción que tenga que realizar. Con esta prueba queda demostrado el correcto funcionamiento de la interfaz incluso fuera de la ejecución de la terapia.

Por último, se muestran otras animaciones en las que está relacionado el movimiento del robot junto con la animación de la interfaz. Estas animaciones pueden ser por ejemplo *SitSleep* y *SitWakeUp:* que para el caso de la ejecución del robot REEM-C se visualiza la acción realizada por el robot y la ejecución de la interfaz, situación que no se da en el robot REEM, ya que al no tener la posibilidad de sentarse, únicamente muestra la animación de la interfaz.

Con esta experimentación se observa que en general la fluidez de la ejecución de las animaciones en correcta ya que cada uno de los robots realiza una planificación de los movimientos para conseguir la animación final. Sin embargo, en animaciones

que requieren un ritmo mas rápido de ejecución, como es el caso de la animación *dance_macarena* que debe realizarse al ritmo del audio, se puede notar cierta brusquedad al realizar algunos movimientos. Se observa además que existe una pequeña desincronización al realizar el baile de la Macarena, una de las causas posibles es que el tiempo de simulación en Gazeno no es el mismo al tiempo real.

Para concluir, indicar que el conjunto de las animaciones diseñadas y mostradas en esta experimentación, son las utilizadas en las terapias de rehabilitación original con el robot NAO, por lo que se demuestra la exitosa integración de las mismas en los nuevos componentes teniendo en cuenta las restricciones morfológicas de cada robot. Indicar también que se deja analizado el procedimiento para la realización de las animaciones para futuras mejoras o nuevos proyectos.

Text-to-Speech

Para realizar la prueba de esta funcionalidad se usa un texto para el cual no existe un fichero de audio correspondiente, de tal manera que el componente hará uso de la funcionalidad *Text-to-Speech* para reproducirlo.

El resultado de la experimentación se puede observar en el vídeo **Text-to-Speech - Evaluación REEM / REEM-C**. Como se muestra, la simulación para ambos robots se realiza correctamente y además con el mismo resultado, ya que se tiene un único desarrollo que es válido para ambos robots.

Con esta experimentación junto con la prueba de las animaciones, se demuestra que queda cumplido el objetivo en relación planteado en el Capítulo 1 de este mismo documento.

5.4. Sesión de la terapia

Revisadas cada una de las funcionalidades por separado y comprobado el correcto funcionamiento de todas, en esta sección se muestra una ejecución completa de una

sesión real de terapia con cada uno de los robots. Estas dos sesiones han sido grabadas y se pueden visualizar en los vídeos **Sesión terapia - Evaluación REEM** y **Sesión terapia - Evaluación REEM-C** disponibles en el canal de Youtube indicado al inicio del capítulo.

Para cada una de las sesiones de terapia realizadas se ha desarrollado un nuevo problema modificado del dominio real de ejecución de terapias de NAOTherapist. Se han seleccionado un conjunto de 8 poses, con un descanso tras la realización de las 4 primeras poses. En el descanso se observa una interacción del robot con el paciente mediante una animación para que el usuario no pierda la concentración. En cada una de las sesiones se puede observar la funcionalidad completa desarrollada en este trabajo, compatibilidad con la arquitectura NAOTherapist, reproducción de animaciones y ficheros de audio con sonidos que fomentan la interacción con el usuario y la corrección de posturas incorrectamente realizadas mediante el uso de la interfaz y el *retargeting*.

Comparando los dos componentes se observa que realizan su funcionalidad correctamente aunque es necesario indicar que cabe la posibilidad de que el robot REEM-C, al no ser tan estable como el robot REEM, requiera de ciertas modificaciones en su función de *retargeting* para poder llevar a cabo la sesión de terapia con el robot físico. Esto no se ha podido comprobar en este trabajo dado que no se disponía de ninguno de los robots físicos.

Adicionalmente, tras la realización de todas las experimentaciones se incluye en esta sección una matriz de trazabilidad de los requisitos funcionalies y no funcionales que han sido completados exitosamente durante la realización del proyecto. Se clasifica cada uno de los requisitos por *Realizado, Parcial, No realizado.* indicando como *Realizado* los requisitos que se han cumplido en su totalidad, *Parcial* aquellos es posible mejorar y *No realizado* aquellos requisitos que no se han podido tener en cuenta.

En la Tabla 5.1 se puede ver dicha matriz para los requisitos funcionales. Se puede observar que todos han sido conseguidos a excepción de dos que se encuentran con posibilidades de mejora. Estos dos requisitos son los relacionados con la función de

retargeting que como ya se ha comentado en este documento ha ocasionado bastantes dificultades durante el desarrollo de este proyecto. Dada la complejidad de la función de *retargeting* se ha comprobado que es posible realizar un 70 % de las poses requeridas por la arquitectura inicial siendo posible mejorar algunas posturas relacionadas con la articulación del giro del codo.

	REQUISITOS FUNCIONALES		
	Realizado	Parcial	No realizado
RF-01	✓		
RF-02	✓		
RF-03	✓		
RF-04	✓		
RF-05		P	
RF-06	✓		
RF-07	✓		
RF-08	✓		
RF-09	✓		
RF-10	✓		
RF-11		P	
RF-12	✓		
RF-13	✓		
RF-14	✓		
RF-15	✓		

Tabla 5.1: Matriz de requisitos funcionales realizados.

En la Tabla 5.2 se muestra la matriz para los requisitos no funcionales. Se puede observar que todos los requisitos han sido incluidos en el sistema a excepción del relacionado con el control de colisiones de los robots, que sería posible perfeccionar ya que actualmente si el robot (tanto robot REEM como REEM-C) estima que se va a producir colisión al realizar la postura, no la realiza.

	REQUISITOS NO FUNCIONALES		
	Realizado	Parcial	No realizado
RNF-01	✓		
RNF-02	✓		
RNF-03	✓		
RNF-04	✓		
RNF-05	✓		
RNF-06	✓		
RNF-07	✓		
RNF-08	✓		
RNF-09		P	
RNF-10	✓		
RNF-11	✓		
RNF-12	✓		
RNF-13	✓		
RNF-14	✓		
RNF-15	✓		
RNF-16	✓		
RNF-17	✓		
RNF-18	✓		

Tabla 5.2: Matriz de requisitos no funcionales realizados

Para concluir, con esta última experimentación se engloba al resto de pruebas realizadas que junto con la matriz de requisitos conseguidos se demuestra que los dos nuevos componentes han sido integrados exitosamente en la arquitectura original NAOTherapist quedando cumplidos todos los objetivos señalados para este trabajo.

Capítulo 6

Planificación y presupuesto

En este capítulo se presenta la planificación estimada para la realización del presente proyecto, así como el presupuesto estimado de realización del mismo.

El capítulo consta de tres secciones, indicando en la primera la metodología utilizada para la planificación. A continuación, en la siguiente sección se realiza el detalle de la planificación del proyecto. Por último, se realiza un análisis del coste que llevaría asociado la realización del proyecto

6.1. Metodología de planificación

Para realizar la planificación del proyecto, se ha decidido seguir un modelo en cascada. Dicho modelo ordena las fases del proyecto de forma secuencial, de tal manera que el inicio de una fase debe esperar a la finalización de la fase anterior. Al final de cada fase, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

Se ha tomado esta decisión a la hora de elegir el modelo en cascada porque es un modelo sencillo de implementar y entender, que además esta orientado a la redacción de documentos y promueve una metodología de trabajo efectiva: Definir antes que diseñar,

diseñar antes que codificar.

El modelo en cascada original, tal y como lo plantearon Winston W. Royce en 1970 y Barry Boehm en 1980, consta de las siguientes fases:

- **Análisis del sistema:** en esta fase se analizan las necesidades del usuario final. Para ello, se deben de programar reuniones con el usuario (en este caso, el tutor del proyecto) para conocer exactamente sus necesidades. El objetivo final de esta fase es tener un documento de requisitos que detalle claramente qué tiene que hacer el sistema a desarrollar y cómo tiene que hacerlo.

- **Diseño del sistema:** esta fase descompone y organiza el sistema en los componentes que se puedan elaborar por separado. Además se especifica que debe hacer o contener exactamente cada uno de los componentes que se definan para que al terminar la fase se conozcan claramente estos y poder realizar la posterior implementación.

- **Implementación del sistema:** en base al diseño del sistema, se desarrolla la implementación del sistema

- **Pruebas del sistema:** en esta fase el desarrollador realiza las pruebas necesarias para comprobar el correcto funcionamiento del sistema y que cumpla con todos los requisitos especificados por el usuario.

- **Verificación y mantenimiento:** Por último el usuario final comprueba que el resultado obtenido es el deseado y satisface sus necesidades. Durante esta fase, se mantiene el sistema en activo para realizar las posibles modificaciones del sistema.

Además de las fases del modelo en cascada, se considera necesario incluir una fase más de ***Documentación*** para elaborar la documentación del proyecto. Esta fase se inicia a la mitad de la fase de *Análisis del sistema*, por lo que es la excepción del modelo en cascada establecido, pero una fase muy necesaria. En la Figura 6.1 se muestra un diagrama de ejemplo de la metodología en cascada.

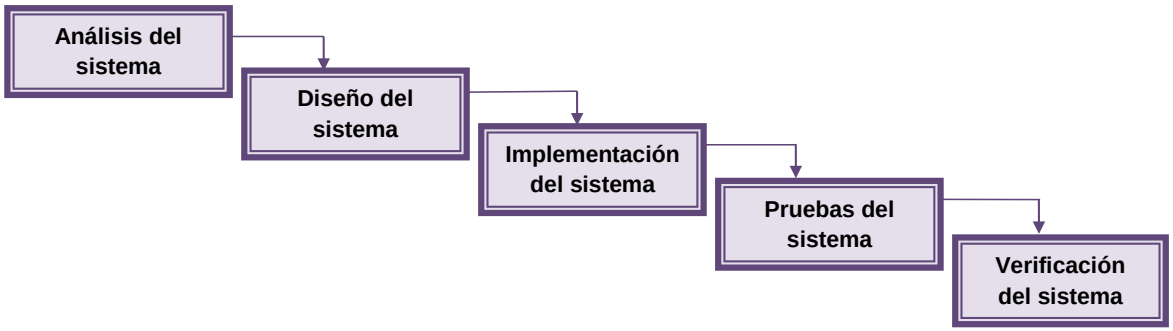


Figura 6.1: Modelo de planificación en cascada.

Comentar que dada la extensión de la fase de implementación, se ha utilizado la metodología basada en prototipos (ver Sección 3.5) para la correcta finalización de la fase.

6.2. Planificación

En este apartado se incluye la planificación realizada para el proyecto, incluyendo las fases y tareas identificadas en el mismo. Además se realizará una estimación de las horas de dedicación por semana.

El proyecto tiene una duración total de un año, comenzando el día 1 de febrero de 2015 y finalizando el 2 de junio de 2016. En la Tabla 6.1 se indican cada una de las fases definidas, detallando, a su vez, las principales tareas que componen cada una de estas, junto con la duración estimada de cada una.

En lo referente a las tareas críticas, destacar una tarea que es completamente necesaria para poder continuar con el trabajo, que es la tarea de *Implementación retargeting*. Si no se consigue una correcta implementación del *retargeting* no tiene sentido continuar con el desarrollo del proyecto. De ahí que sea una de las tareas a la que se le estima un mayor tiempo de dedicación (Tabla 6.2). Además de esta tarea, son muy importantes las tareas de *Desarrollo del interfaz* y *Reproducción de audio* ya que sin ellas

TAREAS	DURACIÓN (DÍAS)	FECHA INICIO	FECHA FIN
Análisis del sistema	**60**	**01/02/2015**	**01/04/2015**
Definición de los objetivos del proyecto	7	01/02/2015	07/02/2015
Lectura documentación ROS	13	08/02/2015	20/02/2015
Lectura documentación Gazebo	6	21/02/2015	26/02/2015
Lectura documentación REEM y REEM-C	6	27/02/2015	04/03/2015
Lectura documentación NAOTherapist/RoboComp	6	05/03/2015	10/03/2015
Especificación de requisitos	12	11/03/2015	22/03/2015
Diseño casos de uso	9	23/03/2015	01/04/2015
Diseño del sistema	**30**	**02/04/2015**	**01/05/2015**
Definición de la arquitectura del sistema	13	02/04/2015	14/04/2015
Componente ReemComp y ReemCompC	7	15/04/2015	21/04/2015
Interfaz de usuario	10	22/04/2015	01/05/2015
Implementación del sistema	**315**	**02/05/2015**	**21/03/2016**
Integración de ROS en NAOTherapist	20	02/05/2015	22/05/2015
Comprobación conexión	5	23/05/2015	28/05/2015
Comandos simples con Python	20	29/05/2015	18/06/2015
Implementación retargeting	120	19/06/2015	17/10/2015
Desarrollo interfaz	15	18/10/2015	02/11/2015
Comunicación componente con la interfaz	15	03/11/2015	18/11/2015
Interfaces adicionales	20	19/11/2015	09/12/2015
Desarrollo animaciones	20	10/12/2015	30/12/2015
Integración REEM-C	70	31/12/2015	10/03/2016
Manejo de excepciones y scripting	10	11/03/2016	21/03/2016
Pruebas del sistema	**30**	**22/03/2016**	**22/04/2016**
Definición entorno pruebas THERAPIST	15	22/03/2016	06/04/2016
Ejecución pruebas	15	07/04/2016	22/04/2016
Verificación del sistema	**10**	**23/04/2016**	**02/05/2016**
Documentación	**450**	**10/03/2015**	**02/06/2016**

Tabla 6.1: Planificación del trabajo de fin de grado

sería imposible la integración total del robot REEM con el proyecto NAOTherapist.

Una vez indicadas las principales fases y tareas del proyecto, junto con la identificación de las tareas críticas del mismo, se incluye la secuenciación y temporización de cada una de ellas mediante un diagrama de Gantt (Figura 6.2) que se ha elaborado con la herramienta *Gantt project*[1] en su versión 2.7.1 para Windows.

[1] http://www.ganttproject.biz/

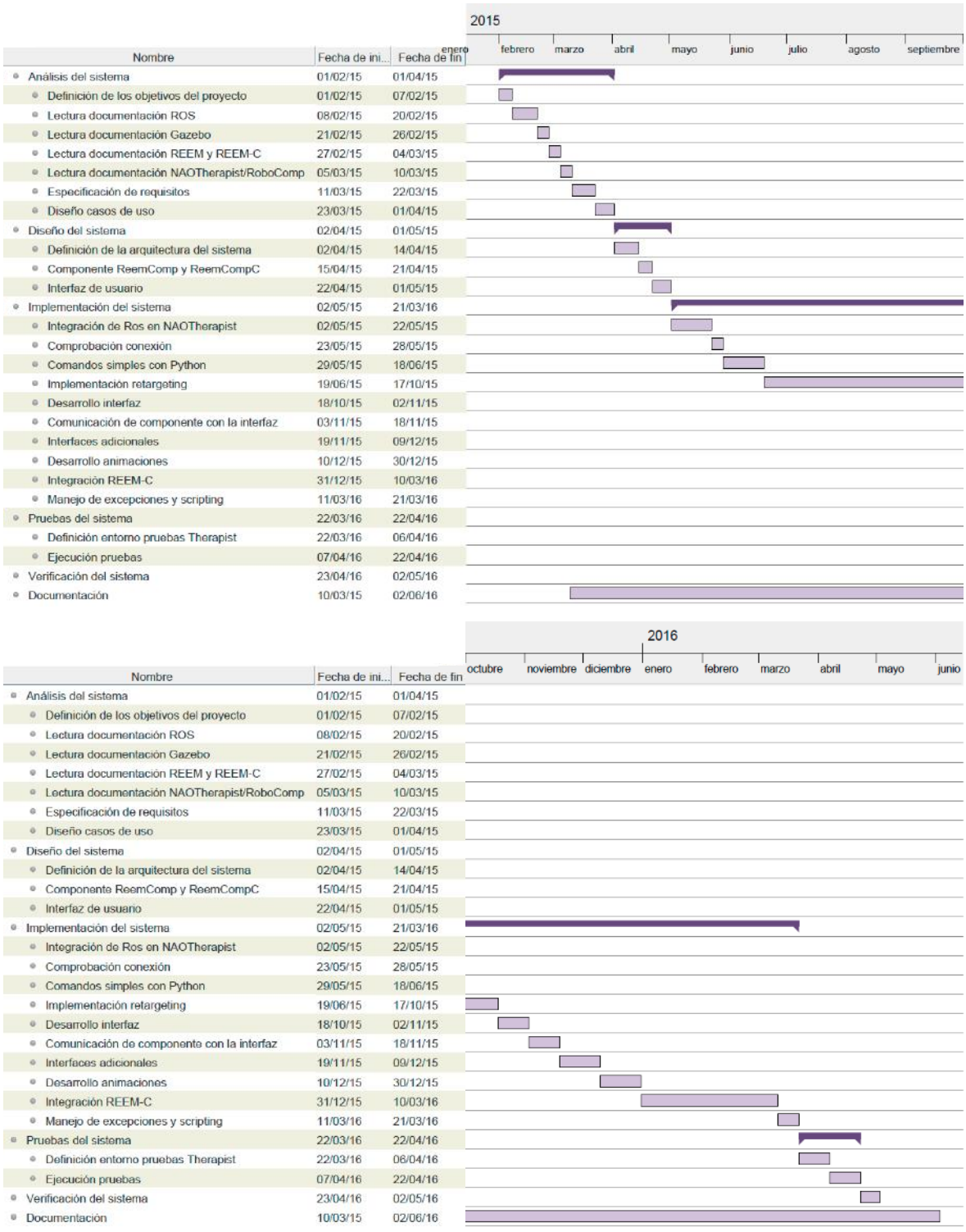
Como se puede observar en la Figura 6.2, se incluyen las fases y tareas que se indicaron anteriormente en la Tabla 6.1, especificando la fecha de inicio y fin de cada una de ellas. Con este diagrama se puede ver claramente el modelo en cascada de planificación, ya que hasta que no termina una fase o tarea no se inicia la siguiente, a excepción de la fase de documentación que es independiente a este modelo.

Habiendo establecido toda la planificación del presente proyecto, se ha elaborado una estimación de tiempo de dedicación para cada uno de los meses de trabajo del proyecto. Esta información se puede ver en la Tabla 6.2

Mes	Horas / Semana	Total horas
Febrero 2015	10	40
Marzo 2015	5	20
Abril 2015	5	20
Mayo 2015	10	40
Junio 2015	10	40
Julio 2015	10	40
Agosto 2015	5	20
Septiembre 2015	10	40
Octubre 2015	5	20
Noviembre 2015	10	40
Diciembre 2015	5	20
Enero 2016	5	20
Febrero 2016	5	20
Marzo 2016	10	40
Abril 2016	10	40
Mayo 2016	10	40
Total	**125**	**500**

Tabla 6.2: Horas de dedicación por semana

En dicha tabla se puede observar que el trabajo se ha repartido bastante a lo largo de los meses de duración del proyecto. Se puede observar también, que las horas de dedicación estimadas para la realización de este proyecto son 500 horas.



Nombre	Fecha de ini...	Fecha de fin
Análisis del sistema	01/02/15	01/04/15
Definición de los objetivos del proyecto	01/02/15	07/02/15
Lectura documentación ROS	08/02/15	20/02/15
Lectura documentación Gazebo	21/02/15	26/02/15
Lectura documentación REEM y REEM-C	27/02/15	04/03/15
Lectura documentación NAOTherapist/RoboComp	05/03/15	10/03/15
Especificación de requisitos	11/03/15	22/03/15
Diseño casos de uso	23/03/15	01/04/15
Diseño del sistema	02/04/15	01/05/15
Definición de la arquitectura del sistema	02/04/15	14/04/15
Componente ReemComp y ReemCompC	15/04/15	21/04/15
Interfaz de usuario	22/04/15	01/05/15
Implementación del sistema	02/05/15	21/03/16
Integración de Ros en NAOTherapist	02/05/15	22/05/15
Comprobación conexión	23/05/15	28/05/15
Comandos simples con Python	29/05/15	18/06/15
Implementación retargeting	19/06/15	17/10/15
Desarrollo interfaz	18/10/15	02/11/15
Comunicación de componente con la interfaz	03/11/15	18/11/15
Interfaces adicionales	19/11/15	09/12/15
Desarrollo animaciones	10/12/15	30/12/15
Integración REEM-C	31/12/15	10/03/16
Manejo de excepciones y scripting	11/03/16	21/03/16
Pruebas del sistema	22/03/16	22/04/16
Definición entorno pruebas Therapist	22/03/16	06/04/16
Ejecución pruebas	07/04/16	22/04/16
Verificación del sistema	23/04/16	02/05/16
Documentación	10/03/15	02/06/16

Figura 6.2: Diagrama de Gantt

6.3. Presupuesto

En este apartado se elabora el presupuesto del presente Trabajo de Fin de Grado. Para determinar el coste total de desarrollo se han tenido en cuenta los diferentes tipos de costes en los que se ha incurrido durante la realización el mismo, así como el tiempo de dedicación expuesto en el apartado de planificación anterior. Los puntos a tener en cuenta para llevar a cabo el análisis económico son los siguientes:

- Costes de personal
- Costes de material
- Costes indirectos

Para menor complejidad a la hora de realizar los cálculos se supone que es una empresa propia la que ha llevado a cabo toda la realización del proyecto. La unidad monetaria que se utiliza es el Euro (€), teniendo en cuenta un redondeo, cuando proceda, de dos decimales.

Por último, para calcular el coste total del proyecto se tienen en cuenta todos los costes, impuestos aplicables y el porcentaje de ganancia deseado ligado a los riegos asumidos.

6.3.1. Coste de personal

En este apartado se detalla el importe del presupuesto dedicado a los costes de personal. Para realizar este calculo se tiene en cuenta las horas de dedicación y el rol asignado para la realización de todas las tareas. Los roles participantes en el presente proyecto son:

- **Jefe de proyecto:** Es la persona encargada de dirigir y coordinar el proyecto. Este rol estará presente a lo largo de todo el proyecto.

- **Analista / Diseñador:** Es la persona encargada de la elaboración de los requisitos y diseño del sistema.
- **Programador:** Es quien realizará la implementación del sistema que anteriormente ha diseñado el analista.
- **Técnico de pruebas:** Su función será la de realizar las pruebas y validación del sistema.

Habiendo definido los roles participantes del proyecto, se muestra en la Tabla 6.3 el presupuesto para estos según los meses de dedicación en el proyecto. Indicar que, el salario al mes para cada rol ha sido establecidos según la tabla salarial publicada en el BOE[2], nº 80 del 2 de abril de 2014, sección III, página 28267 teniendo en cuenta los posibles complmentos adicionales para cada rol.

Rol	Meses Trabajados	Salario mes (€)	Presupuesto Proyecto (€)
Analista/Diseñador	3	1.490,03	4.470,09
Programador	10,7	1.322,17	1.414,22
Técnico de pruebas	1,3	1.132,02	1.471,63
Jefe de proyecto	15	1.657,85	24.867,75
		TOTAL	44.956,69

Tabla 6.3: Presupuesto según meses trabajados y salario base

Por tanto, como se observa en la Tabla 6.3 el importe del coste de personal es **44.956,69 euros**

[2]https://boe.es/boe/dias/2014/04/02/pdfs/BOE-A-2014-3544.pdf

6.3.2. Costes materiales

En esta sección se indican los costes de material, teniendo en cuenta el producto, precio, periodo de amortización del producto, el uso en meses y el coste final de cada producto. Según todas estas variables, el coste total de cada producto se calcula de la siguiente como $coste = \frac{precio}{periodo\ amortización}$

Producto	Precio (€)	Periodo Amortización (Meses)	Uso (Meses)	Coste proyecto (€)
Portatil Lenovo Y50-70	1.000	48	15	312,50
Kinect XBOX 360	150	48	15	46,88
Simulador Gazebo	0	24	15	0
Framework ROS	0	24	15	0
Licencia Office Professional	539	60	15	134,75
Adobe Acrobat XI Pro	676	60	15	169,00
Licencia Lucidchart	0	60	6	0
Licencia Microsoft Visual Studio	600	60	6	60
Licencia Gantt Project	0	60	1	0
Licencia ShareLatex	0	60	15	0
			TOTAL	723,13

Tabla 6.4: Coste del material

En la Tabla 6.4 se observa que el total de los costes materiales asciende a **723,13 euros**, siendo el portátil el producto más caro, pero indispensable para la realización del proyecto.

6.3.3. Costes indirectos

En esta sección se indica el cálculo de los costes indirectos de este proyecto. Se asume que los costes indirectos son los correspondientes al gasto de luz, teléfono, internet, reparaciones,... y se establece como un 5 % aplicado sobre los gastos directos (costes de personal y costes de material). Se puede observar en la Tabla 6.5 que el importe de los costes indirectos es de **2.283,99 euros**. Es un coste algo elevado pero se tiene en cuenta la larga duración del proyecto.

	IMPORTE
Coste de personal	44.956,69
Coste materiales	723,13
TOTAL COSTES DIRECTOS	**45.679,81**
Costes indirectos (5%)	**2.283,99**

Tabla 6.5: Costes indirectos

6.3.4. Costes totales

Por último, tras el cálculo de todos los tipos de costes establecidos, en esta sección se realiza el cálculo del coste total del proyecto. Se asume que el impuesto del valor añadido (IVA) ya esta incluido en todos los importes indicados hasta el momento, por lo que únicamente resta añadir un 10 % en conceptos de beneficios de la realización de este proyecto.

En la Tabla 6.6 se puede observar que el **presupuesto total** estimado de este proyecto es de **CINCUENTA Y DOS MIL SETECIENTOS SESENTA CON DIECIOCHO EUROS** y se estima obtener un beneficio de **4.793,38 euros**

Descripción	Importe
Coste de personal	44.956,69
Coste materiales	723,13
Costes indirectos (5%)	2.283,99
IMPORTE TOTAL SIN BENEFICIOS	**47.963,80**
Beneficio (10%)	4.793,38
PRESUPUESTO TOTAL	**52.760,18**

Tabla 6.6: Presupuesto total

Capítulo 7

Conclusiones y trabajos futuros

En este último capítulo se exponen las conclusiones finales tras el desarrollo del trabajo. También se detallan las posibles líneas de investigación futuras y posible mejoras que podrían aplicarse al sistema

7.1. Discusión

Con el desarrollo de este trabajo se ha logrado establecer una integración completa de dos nuevos componentes en la arquitectura NAOTherapist. Estos dos nuevos componentes ReemComp y ReemCompC pueden sustituir al componente NAORobot de la arquitectura original haciendo que se puedan utilizar el robot REEM o el robot REEM-C en vez del robot NAO y cumpliendo la misma funcionalidad que este. Por lo tanto, con esta integración es posible utilizar cualquiera de estos tres robots para llevar a cabo las sesiones de terapias con niños.

Además se ha desarrollado una interfaz gráfica y unas funciones de reproducción de audio para potenciar la interacción con el usuario. Gracias a la interfaz se le puede mostrar al usuario un resultado de la postura que esta tomando al realizar la terapia y con las distintas reproducciones de audio orientarle en la terapia y animarle en la

realización de la misma.

Por otro lado, se ha logrado desarrollar una función de transformación de la arquitectura de un esqueleto humano proporcionada por un sensor 3D Kinect a la arquitectura del robot a pesar de las distintas dificultades inesperadas que han surgido a lo largo del desarrollo del trabajo. Entre estas dificultades es necesario comentar la desviación que se observa en la articulación de apertura del hombro de los nuevos robots que no se tenía en el robot NAO original y que hace que la función desarrollada tenga posibilidades de mejora para algunas de las posturas de la colección de la arquitectura NAOTherapist. Dado que la función ha sido desarrollada en base al modelo del robot NAO, es posible que esta mejora pudiera llevarse a cabo eligiendo otros planos de referencia para desarrollar la traducción de ángulos.

Para concluir, el objetivo principal y final de este proyecto que era conseguir una integración de los robots REEM y REEM-C en la arquitectura NAOTherapist ha sido logrado exitosamente con el desarrollo del trabajo.

7.2. Conclusiones técnicas

Con el desarrollo del proyecto se obtienen diversas conclusiones para cada una de las funcionalidades integradas pero haciendo incapié en el hecho de que se han logrado los objetivos iniciales.

1. Se ha estudiado, analizado y comprendido el funcionamiento de la arquitectura del proyecto de investigación NAOTherapist. Posteriormente se ha integrado exitosamente el *framework* robótico ROS (*Robot Operating System*) junto con el *framework* RoboComp (framework utilizado en el proyecto NAOTherapist). Con esta integración es posible realizar la ejecución de la arquitectura NAOTherapist teniendo en cuentra la funcionalidad de ROS.

2. Se ha logrado conocer la ejecución de movimientos y posturas simples de los ro-

bots en el simulador Gazebo. Por ello, se ha conseguido conocer la forma de enviar tareas al robot a través del nuevo componente desarrollado en función de los requerimientos de la arquitectura de NAOTherapist. Además al conocer la posibilidad de movimiento del robot REEM, se han desarrollado ciertas animaciones que se utilizan en la arquitectura original, dotando al robot REEM y REEM-C de las mismas habilidades (teniendo en cuenta las características de cada robot) de interacción que el robot NAO.

3. Se ha desarrollado una función de *retargeting* que es capaz de transformar los ángulos de las articulaciones de un esqueleto humano captado con una Kinect a las articulaciones del robot, consiguiendo de este modo los mismos movimientos en el robot que pueda realizar el humano. Este desarrollo se ha logrado con una similitud bastante elevada en comparación con la misma función en el robot original.

4. Para fomentar la interacción humano-robot y simular la misma funcionalidad disponible en la arquitectura con el robot NAO, se ha desarrollado una interfaz gráfica que simula los leds y botones físicos que dispone el robot NAO en su cuerpo. Adicionalmente, se ha incluido una nueva funcionalidad para que la interfaz pueda ser configurable y sea el usuario el que elija los colores y formas con los que desea realizar la terapia. Del mismo modo, se ha desarrollado un sistema de reproducción de audio y *text-to-Speech* que permiten la comunicación del robot con el usuario.

5. Después de todo el desarrollo del trabajo se ha conseguido la integración completa del robot REEM en la arquitectura NAOTherapist permitiendo que sea este el que realice las sesiones de terapia. Pese a no haber sido probado en un entorno con el robot físico, por el buen funcionamiento del sistema en el simulador y la estabilidad de este robot, se puede afirmar que funcionaría correctamente con un robot REEM real.

6. Gracias conocimiento adquirido durante al desarrollo del trabajo con el robot REEM y un análisis posterior realizado, se ha conseguido la integración del robot REEM-C,

siguiente versión del robot REEM, en la arquitectura NAOTherapist de modo que que sea este el que realice las sesiones de terapia del mismo modo que su antecesor. Debido a los resultados obtenidos y a la posible inestabilidad del robot por el equilibrio de las piernas, es posible que para la realización de las terapias con el robot físico fuera necesarias ciertas modificaciones en el componente pese a que en una simulación virtual la sesión de terapia sea correcta.

7.3. Líneas futuras

Finalizada la realización de este trabajo, tras analizar las posibilidades y capacidades que ofrecen los dos nuevos robots integrados surgen diversas ideas de futuras líneas de investigación que serían favorables para la mejora de la arquitectura y de los nuevos componentes.

Se ha desarrollado en este trabajo una función de *retargeting* con buenos resultados que permite trasformar la postura de un usuario en ambos robots. Pero esta función no se ha logrado al 100 % debido a los diversas dificultades encontradas en las articulaciones de los nuevos robots. Por ello sería interesante realizar una nueva investigación de mejora sobre esta función por ejemplo eligiendo otros planos para realizar la conversión de los ángulos dado que los tratados en este proyecto se han tomado teniendo en cuenta el modelo base del robot NAO original.

Con el desarrollo de este proyecto se han implementado ciertas animaciones que se tenían definidas en la arquitectura original como es por ejemplo el baile de la Macarena. Dado que ha quedado definido el proceso de desarrollo de las animaciones sería interesante incluir más animaciones o bailes que proporcionaran una mayor interacción a la terapia e hicieran más entretenidos los descansos. Un ejemplo sería el desarrollo de la animación de taichi que el robot NAO tiene predefinida de serie.

En relación a la interfaz, al ser un componente que no depende de la morfología del robot como sucede con el robot NAO, la capacidad de mejora es altísima pudiendo

añadir contenido, modificar e intercambiar en ejecución. Una posibilidad sería añadir más figuras para el cambio de modelo de interfaz, incluir un vídeo en el momento que el robot realiza un baile como el de la Macarena, añadir emoticonos cuando la postura es correcta, etiquetas o *tooltips* en los botones para facilitar la comprensión de la acción que realiza,... Con estas mejoras se daría mucho valor al componente al tratar de mejorar la interacción humano-robot.

Apéndice A

Manual de instalación

A.1. Instalación de los componentes

En esta sección, se detalla el manual de instalación de todo los componentes necesarios para el funcionamiento del componente NaoComp y los nuevos componente ReemComp y ReemCompC.

Indicar que este manual de instalación se ha desarrollado teniendo en cuenta un sistema operativo Ubuntu 12.04 que hasta día de hoy es la última versión compatible con los nuevos robots.

A.2. Instalación del software necesario para ReemComp/ReemCompC

En esta sección se detallan esquemáticamente los pasos seguidos para la instalación de los componentes necesarios para el funcionamiento del nuevo componente ReemComp y ReemCompC en la arquitectura NAOTherapist

A.2.1. Instalación *framework* robótico ROS

Dados los requisitos de software del robot REEM y REEM-C, se detalla la instalación de la versión ROS Hydro[1] ya que es el software requerido para su funcionamiento

1. Configurar los repositorios de Ubuntu siguiendo los pasos de la guía de Ubuntu[2]. Es necesario activar los componentes *Universe / Multiverse*. Se pueden activar en:

```
Menu principal: Sistema > Administrador > Fuentes de Software
```

2. Configurar `sources.list`

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
precise main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Configuración de claves requeridas para establecer conexión con el módulo principal de ROS

```
wget \url{https://raw.githubusercontent.com/ros/rosdistro/master/ros.key}
-O - | sudo apt-key add -
```

4. Instalación

   a) Actualizar el índice de paquetes general del sistema

   ```
   sudo apt-get update
   ```

   b) Instalación completa del *framework* ROS Hydro:

   ```
   sudo apt-get install ros-hydro-desktop-full
   ```

5. Inicializar `rosdep`, necesario para poder ejecutar algunos de los principales componentes de ROS:

```
sudo rosdep init
rosdep update
```

[1] http://wiki.ros.org/hydro/Installation/Ubuntu (Accedida el 27/01/2016)

[2] https://help.ubuntu.com/community/Repositories/Ubuntu Accedido por última vez el 11/01/2016

6. Configuración de las variables de entorno

```
echo "source /opt/ros/hydro/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

7. Instalación de la herramienta `rosinstall`. Herramienta de línea de comandos que permite descargar código fuente de ROS.

```
sudo apt-get install python-rosinstall
```

A.2.2. Integración del modelo del robot REEM para Gazebo

A continuación se detallan los pasos para la instalación del robot REEM junto con el simulador Gazebo[3]

1. Crear workspace para el robot REEM, directorio principal que contiene todas las herramientas de REEM.

```
mkdir -p ~/reem-sim_ws/src \&\& cd ~/reem-sim_ws/src
catkin_init_workspace
```

2. Descargar archivos de instalación del modelo del robot REEM para Gazebo:

```
wstool init .
wstool merge https://raw.githubusercontent.com/pal-robotics/pal-ros-pkg/master/
reem-sim-hydro.rosinstall
```

3. Actualización de los paquetes del robot REEM descargados:

```
wstool update -j8
```

4. Instalación de los paquetes de ROS necesarios para el correcto funcionamiento de REEM:

```
cd ..
rosdep install --from-paths src --ignore-src --rosdistro hydro -y
sudo apt-get install ros-hydro-ros-control ros-hydro-ros-controllers
```

[3]http://wiki.ros.org/Robots/REEM/Tutorials/Launching%20a%20REEM%20Gazebo%20simulation Accedido por última vez 20/04/2016

5. Compilación de los paquetes descargados y contenidos en el workspace del robot REEM:

```
catkin_make
```

A.2.3. Integración del modelo del robot REEM-C para Gazebo

A continuación se detallan los pasos para la instalación del robot REEM-C junto con el simulador Gazebo[4]

1. Crear workspace para el robot REEM-C, directorio principal que contiene todas las herramientas de REEM-C.

```
mkdir -p ~/reemc-sim_ws/src \&\& cd ~/reemc-sim_ws/src
catkin_init_workspace
```

2. Descargar archivos de instalación del modelo del robot REEM para Gazebo:

```
wstool init .
wstool merge \
  https://raw.github.com/pal-robotics/pal-ros-pkg/master/reemc-sim-hydro.rosinsta
```

3. Actualización de los paquetes del robot REEM descargados:

```
wstool update -j8
```

4. Instalación de los paquetes de ROS necesarios para el correcto funcionamiento de REEM:

```
cd ..
rosdep install --from-paths src --ignore-src --rosdistro hydro -y
```

5. Compilación de los paquetes descargados y contenidos en el workspace del robot REEM:

```
catkin_make
```

[4]http://wiki.ros.org/Robots/REEM-C Accedido por última vez 20/04/2016

A.2.4. Configuración variables de entorno

Comprobar que se ha añadido lo siguiente en el fichero /.bashrc. Si no es así, añadir información restante.

```
source /opt/ros/hydro/setup.bash
source ~/reem-sim_ws/devel/setup.bash
source ~/reemc-sim_ws/devel/setup.bash
```

Comprobar funcionamiento A continuación se indica las sentencias que hay que ejecutar para poder iniciar el simulador con cada uno de los robots.

A.2.4.1. Robot REEM

Para probar el correcto funcionamiento del simulador con el robot REEM ejecutar en una terminal la siguiente sentencia y se abrirá el simulador Gazebo con el robot REEM:

```
roslaunch reem_gazebo reem_empty_world.launch
```

A.2.4.2. Robot REEM-C

Para probar el correcto funcionamiento del simulador con el robot REEM-C ejecutar en una terminal la siguiente sentencia y se abrirá el simulador Gazebo con el robot REEMC:

```
roslaunch reemc_gazebo reemc_empty_world.launch
```

Adicionalmente, en el caso del robot REEM-C para poder realizar los movimientos es necesario cargar el controlador de las trayectorias de las articulaciones por lo que se debe ejecutar en una consola independiente la siguiente sentencia:

```
roslaunch reemc_controller_configuration joint_trajectory_controllers.launch
```

A.3. Instalación NAOTherapist

La arquitectura NAOTherapist actualmente es compatible con los sistemas operativos Ubuntu 12.04 y Ubuntu 14, pero dado que para el presente proyecto, como ya se ha indicado, se utiliza la versión Ubuntu 12.04 por las restricciones de los robots, en esta sección se describe la instalación de la arquitectura para dicha versión.

1. Instalar dependencias de NAOTherapist (*javabridge* y *python-weka-wrapper* necesarios para uso de weka en python):

```
sudo apt-get install git default-jdk yakuake zeroc-ice35 python-dev python-pip
sudo apt-get install python-pygame
sudo pip install psutil pyglet pyttsx javabridge python-weka-wrapper
```

2. Descargar e instalar *choregraphe-suite-1.14.5* y *choregraphe-suite-2.1.0*[5] necesarias ambas versiones ya que son las utilizadas por los robots NAO del laboratorio PLG donde se ha desarollado el proyecto.

   a) Copiar el script *Choregraphe* dentro de la carpeta raíz de choregraphe dentro de /bin

   b) Editar el nuevo script para quitarle /bin de la última línea (e incluir el número de licencia como argumento, si se tiene)

   c) Dentro de /lib borrar todos los enlaces simbólicos de libQt* y libsso* (sino el script de yakuake podría dar error de mixed libraries al ejecutar qdbus)

   d) Borrar dentro de esa misma carpeta libpng12.so.0

3. Descargar la arquitectura NAOTherapist al completo

```
git clone \url{https://USUARIOBITBUCKET@bitbucket.org/josgonza/naotherapist.git
~/NaoTherapist}
```

4. Crear `Usuario NaoTherapist` de la siguiente manera. Sirve para que funcionen correctamente los scripts de ejecución de la arquitectura:

[5] https://community.aldebaran-robotics.com/

```
sudo ln -s ~/ /home/NaoTherapist
```

5. Dentro de la raíz de NAOTherapist, ejecutar `transcodeIces.sh`. Este fichero compilas las interfaces *.ice* y las distribuye entre los componentes de la arquitectura.

6. Editar la versión ice en el fichero `config.icestorm.icebox` del componente NT-VisionComp

7. En algunos sistemas es necesario recompilar el planificador MetricFF dentro de NaoTherapist/Components/PLG/pelea2level/planners/MetricFF con make.

8. Añadir las siguientes variables de entorno al final de /.bashrc

```
export AL_DIR=~/Programas/choregraphe-suite-1.14.5-linux64
#export AL_DIR=~/Programas/choregraphe-suite-2.1.0.19-linux64
#export AL_DIR=~/Programas/choregraphe-suite-2.1.3.3-2-linux64
export LD_LIBRARY_PATH=\$AL_DIR/lib:\$LD_LIBRARY_PATH
export PYTHONPATH=\$AL_DIR/lib:\$PYTHONPATH
export PATH=\$PYTHONPATH:\$AL_DIR/bin:\$PATH
```

9. Cerrar la sesión de Ubuntu e iniciar de nuevo.

A.3.1. Comprobar funcionamiento

Para comprobar el funcionamiento de la arquitectura se ejecuta el script *light.sh*. En este script se encuentran distintas opciones:

- **-r/-c:** con la lertra *-r* se indica que se ejecutará la arquitectura con el robot REEM. Si en vez de la letra *r* se utiliza la letra *c* iniciará el sistema con el robot REEM-C y si no se incluye esta opción, se ejecutará la arquitectura con el robot NAO.

- **-o:** esta opción indica el dominio que se va a ejecutar.

- **-f:** esta opción determina el problema a a ejecutar. En este caso las distintas terapias que se han diseñado.
- **-s:** con esta opción se indica que el sistema se ejecutará en modo simulado, si no se incluye es asume que se ejecutará una sesión real con la Kinect.

Gracias a este script es posible llevar a cabo la ejecución con cualquiera de los tres robots, ya que cada una de las opciones llamará a los componentes y herramientas necesarias para la ejecución del mismo.

Apéndice B

English overview

This appendix provides a summary of the work done in this project.

B.1. Introduction

This section describes the context, structure and main goals of the project. The main goal of the project is to develop two new components for the NAOTherapist project [González et al. 2015] using REEM and REEM-C robots. This architecture provides a robot interface (NAO robot in original architecture) wich is able to perform some functions that a therapist usually does in rehabilitation therapies. Specifically, the robot shows to the patients some different poses that the patient must be imitate correctly. The robot can correct the patient pose until it correct. For this project, NAO robot is replaced by REEM and REEM-C robots, maintaining the functionality of the original architecture. This results in two new components, ReemComp and ReemCompC, wich they can replace NAO robot in therapies.

B.1.1. Goals

This section describes the specific objectives of this project:

1. Integration of the ROS framework (*Robot Operating System*) beside the NAOTherapist architecture (which uses RoboComp framework), essential to communicate REEM with other components of the architecture.
2. Make the execution of simple movements with the REEM model in Gazebo simulator through simple commands in Python.
3. Investigate the domain and range of movement of each joint of REEM robot and implementing a transformation function of the angles of the REEM robot using the skeleton model of a human being extracted from the data of a 3D sensor (retargeting). With this retargeting system, the robot can follow the movements of a human using a 3D sensor.

   a) Select the joints of the REEM which allow to do a retargeting.

   b) Transform angles to perform the same movement in REEM then Kinect skeleton.

4. Implement a graphical interface that simulates the functionality of original NAO robot.
5. Integrate the architecture and movement of REEM robot in NAOTherapist system.
6. Integrate the architecture and movement of REEM-C robot in NAOTherapist system based on previous REEM integration.

B.1.2. Structure of the document

The document has been structured starting from the most general content to the most specific. The first chapter corresponds to the introduction, showing a general view of the project, including motivation and main goals. Chapter 2 presents the state of the art, including the current state of the principal elements of the system and an overview of scientific papers related to different aspects of the development. Also, some tools and techniques required for the development of the documents are explained.

Chapter 3 is focused on the analysis and design of the system. This section in-

cludes the rules and restrictions of the system, and all the specification of system requirements and use cases to understand how the system works. Chapter 4 shows the implementation of the system, including the main components developed.

Chapter 5 contains the evaluation of the system. This section explains the results of several experiments and tests that show the behavior of the system and the results obtained.

Chapter 6 contains a development planning of the project, in addition to the budget for this work. Finally, Chapter 7 shows the conclusions and future work for this project.

At the end of this document, two appendix with this english overview and some installation manuals have been added.

This work is complemented with the work performed by Carlos Manzano Carrasco. In spite of the projects have been developed using different domains, both works present some dependencies and confluence points, so there has been collaboration on some points of development. The workload is distributed as it is shown in Table B.1.

B.2. Evaluation

This section describes the evaluation of the system, where there are some experiments that cover all functions programmed in this work. In addition, these tests are used to demonstrate that the goals outlined in Chapter 1 of this document have been accomplished.

All videos of this evaluation are uploaded on the official channel of NAOTherapist project. After finish this work, the new components are included in NAOTherapist architecture increasing system possibilities. Evaluation subsections make reference to this channel:

https://www.youtube.com/playlist?list=PL1Fs33N0K0mZseYF4qz-MLZH0-yyP37o-

TASK	Andrea Haro	Carlos Manzano
System analysys	**50%**	50%
System design	**50%**	50%
System implementation		
Integration of ROS in NAOTherapist	**50%**	50%
Checking conexión ROS	**100%**	0%
Exceptions and scripting	**100%**	0%
New interfaces	**100%**	0%
Simple commands for movements in Python	**50%**	50%
Retargeting	**60%**	40%
Integration REEM-C	**100%**	0%
Interface development	**30%**	70%
Comunication ReempComp-interface	**20%**	80%
Audio play	**0%**	100%
Text-to-Speech	**0%**	100%
System evaluation		
Therapy domain	**100%**	0%
Simon domain	**0%**	100%

Tabla B.1: Workload

B.2.1. Retargeting

In this section, the evaluation of the *retargeting* system is presented. **RetargetingHelper** (see Figure B.1) tool offers a list with the 34 postures included in NAOTherapist architecture, showing the values of each joint in Kinect skeleton. Sending the selected pose, retargeting function is executed and the robot take the specified pose.

10 representative poses from 34 available are been selected to execute the retargeting test, comparing NAO and REEM-C robots. In Figure B.2 the results can be shown:

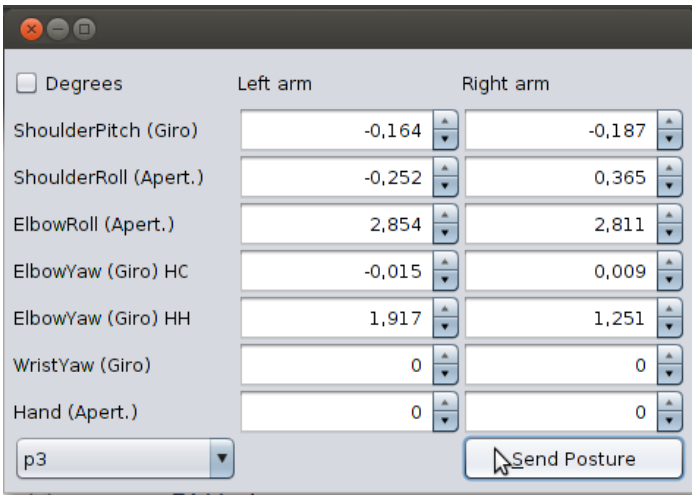


Figura B.1: RetargetingHelper tool.

As can be seen in the different images, there are morphological differences between the two robots. It is noteworthy that despite these differences most poses are pretty close to the original positions of the NAO robot. REEM-C has considerably longer arms in relation with the body, making that the arms are more open in REEM-C than NAO, like it is observed in image (3). The head of REEM is smaller than NAO, so the are differences in image (4) for example.

Moreover, not all problems come from to the morphology of the robot. After many transformations in the elbow, the retargeting could still be improved in positions like (8). When shoulderPitch and elbowYaw angles have a value close to 90 degrees there is a slight deviation in the position, more or less perceived depending of the position of the robot.

After these results it is concluded that the retargeting and trasformation angles works pretty well, so the objectives are demonstrated.

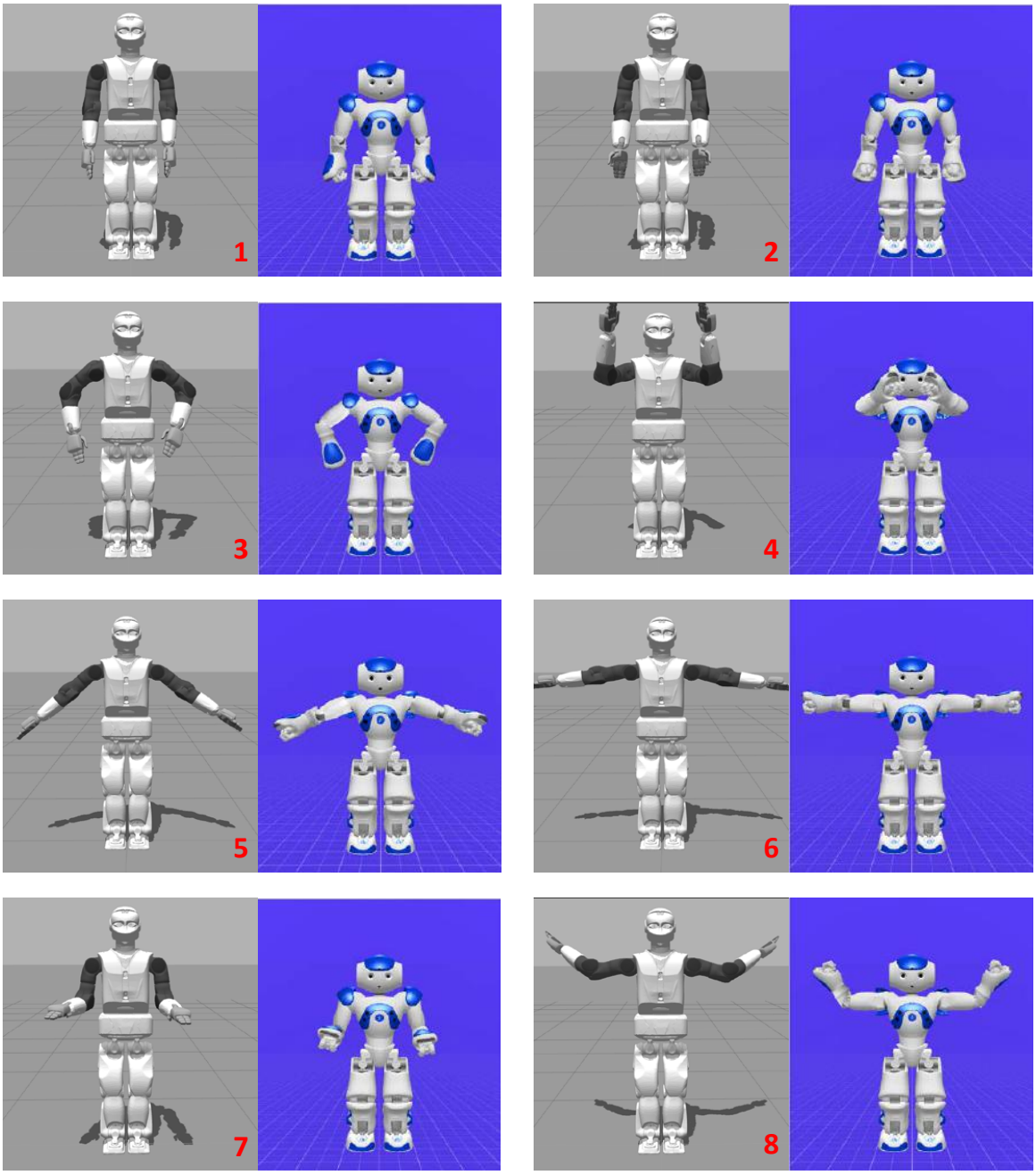


Figura B.2: Pose comparison between REEM-C and NAO robots using retargeting function.

B.2.2. Interface

This section details the experimentation with the graphical interface, responsible for providing feedback to the user.

B.2.3. Posture correction

The original NAO robot uses eye LEDs to provide feedback to the user if his pose is correct. Since the REEM and REEM-C robots they have not any visible LEDs that can facilitate this feedback, an interface has been developed using figures showing an increase in size when the user is closer to the correct pose. This information is independent for each arm, so if you have an arm misplaced and other has the right pose, the interface distinguishes left arm right arm. If the pose is entirely correct the figure shows in his maximum size and if the pose is totally wrong the figure shows the minimum size.

To perform this test indicate that the graphical interface is common for both components. Two different poses has been taken to do this test. The system will check if the position of the user who is in front of the Kinect is the same as the pose of the robot, showing in the interface how close is the user to the correct pose. Figure B.3 shows the results of the experimentation.

The image shows the result of the test, highlighting the independence of each eye, indicating to the user at all time if each arm pose is correct and where is failing. With this experiment, the objectives related to graphical interface are demonstrated.

Buttons and skin changes

To check the functionality asociated to the buttons and demonstrate a proper communication with the rest of the architecture, a test using the *FrontTactil* button is presented (first button from left side on the interface). This button pauses the execution of the system when it is pressed. Pressing the button again, makes the execution to continue. The test shows the trace of the Executive component, which shows the break

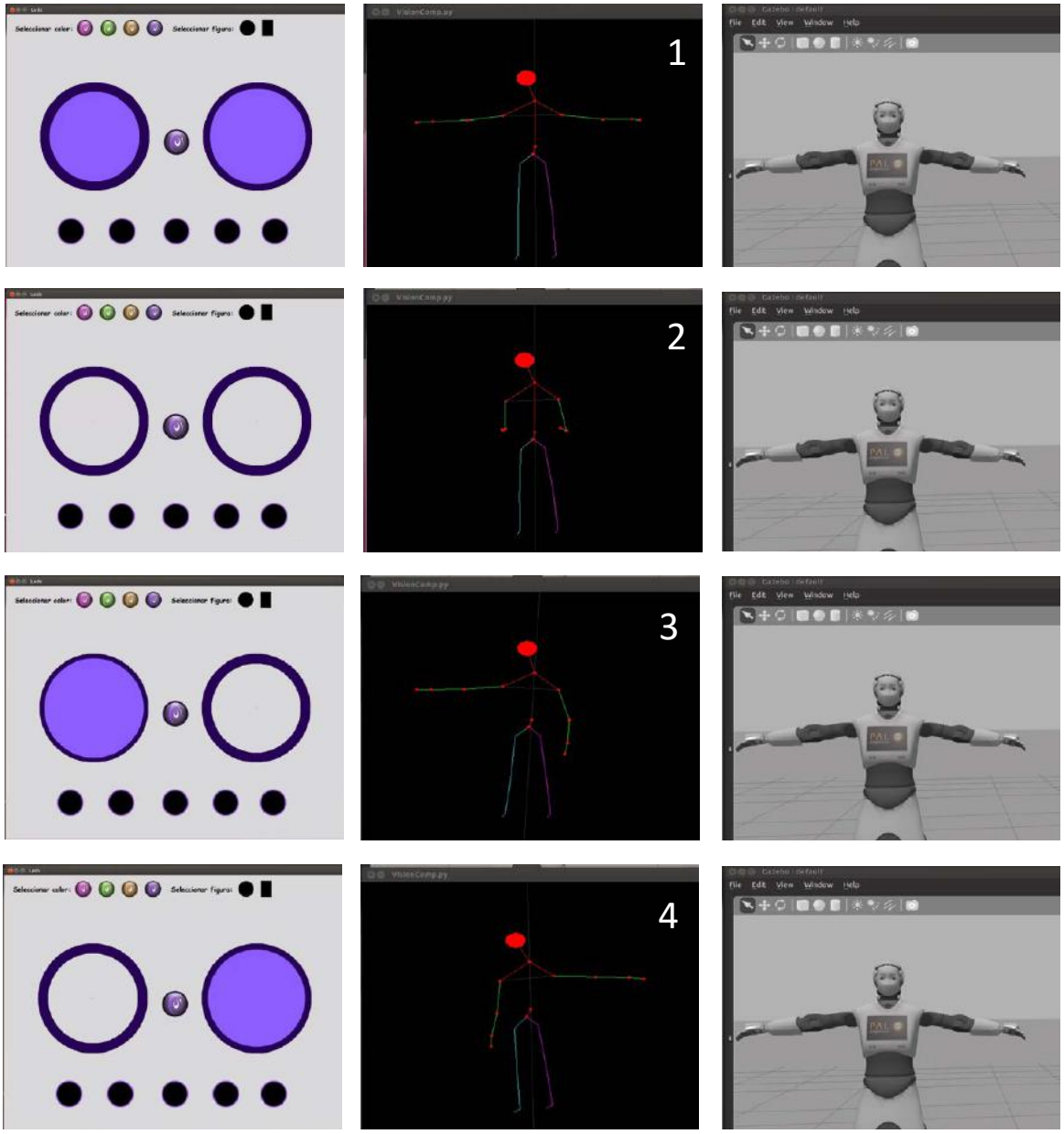


Figura B.3: Evaluation of the pose using graphical interface.

of the session when the button is pressed and resumes the session when it is pressed again.

The following trace is displayed when the button is pressed:

```
Last button pressed set to FrontTactil
->getLastButtonPressed: frontTactil
```

This action informs the Executive component of which button was pressed and executes the corresponding action, which in this case is to pause the execution. The trace for the Executive can be seen below:

```
1 DETECT-PATIENT ['reem', 'pt00']
2 IDENTIFY-PATIENT ['reem', 'pt00']
3 GREET-PATIENT ['reem', 'pt00']
4 START-TRAINING ['reem', 'loc_training', 'pt00', 'loc_show', 'ses10']
Emergency: 0
5 PAUSE-SESSION ['reem', 'pt0', 'ses1']
6 RESUME-SESSION ['reem', 'pt0', 'ses1']
7 INTRODUCE-EXERCISE ['e100r', 'nao', 'pt00', 'ses1']
8 START-EXEERCISE ['e100r', 'nao', 'pt00', 'stand_up']
9 EXECUTE-POSE ['p_id0', 'e100r', 'p6', 't1', 'nao', 'pt00', 'stand_up']
```

This execution of the Executive component is descomposed into low-level actions (actions that have been programmed into the ReemComp component). The pause button is pressed during the execution of the action *START-TRAINING*, causing the session to pause indefinitely. When the same button is pressed again, the action *Resume Session* is executed and the session continues with the correspondant pose.

The interface has different skins who modifies the visual aspect of the interface. Using some buttons presented in the top of the interface, the aspect changes. This buttons works like *FrontTactil* button described previously in this section, with the only difference that the button don't send information to the Executive, because this functionality is unique of REEM robot and there is implemented in his own interface.

With this test, the objectives related to communication between robot interface and the rest of the system are demonstrated.

B.2.4. Animations, audio player and Text-to-Speech

To perform this experiment, an execution sequence is created with all the animations developed to show the correct performance in REEM and REEM-C robots. The results of this experimentation can be shown in **Animaciones - Evaluación REEM / REEM-C** video, available on the YouTube channel mentioned at the beginning of the section. The videos demonstrates that both animations and audio work correctly and they are compatibles one each other, more complex creating animations with sounds integrated.

According to *Text-to-Speech* system, the results can be shown in **Text-to-Speech - Evaluación REEM / REEM-C** video. The video shows that the system is compatible with both robots, fulfilling the stated objective.

B.2.5. Therapist

Reviewed each of the features separately and verified the proper operation of all, this section provides a complete enforcement of a therapy session with REEM and REEM-C robots. These two sessions have been recorded and can be viewed in the **Sesión terapia - Evaluación REEM** and **Sesión terapia - Evaluación REEM-C** videos, available on the Youtube channel indicated at the beginning of the chapter.

For each of the therapy sessions, a new problem based on the actual execution domain of NAOTherapist is developed. They have selected a set of 8 poses divided in two parts, with a break after the completion of the first 4 poses. In the break an interaction is observed with the patient through an animation, so that the user does not lose focus. In each of the sessions the full functionality developed in this work can be seen.

With this test concludes the evaluation section, in which it is shown that all the objectives set at the beginning have been accomplished.

B.3. Conclusions and Future work

With the development of this work it, has been able to establish a full integration of two new components in the NAOTherapist architecture. The conclusions can be summarized has follows:

1. *RoboComp* framework and *ROS* framework are compatible with NAOTherapist architecture, making that the integration with the architecture works properly.

2. Tasks can be sent to the robot through the new component developed according to requirements NAOTherapist architecture. They have developed certain animations that are used in the original architecture and providing REEM and REEM-C of with same functionality found in NAO robot.

3. The *retargeting* allows REEM robot have a similar movement to the original skeleton received from Kinect sensor.

4. The graphical interface communicates with the component created and the rest of the architecture, achieving the main objective by simulating led eyes and physical buttons that the original robot does not have, keeping this functionality through the interface. In addition, it has included a new functionality to change the visual aspect of the interface

5. It has achieved the full integration of REEM robot with the NAOTherapist architecture, allowing the robot make therapy sessions.

6. Thanks to acquired knowledge during development work with REEM robot and a subsequent analysis, it has been possible to make an integration with REEM-C robot, next version of REEM robot in NAOTherapist architecture. This functionality allows to make therapies with both robots properly.

B.4. Future work

After the completion of this work, analyzing the possibilities and capabilities that the two new robots offered, some different ideas of future research that would be favorable for improving the architecture and the new components.

A functional *retargeting* has been made by comparing the position of the robot with the original position of the user, obtaining quite good results. But this feature has not been achieved at 100 % due to various difficulties encountered in the joints of the new robots. It would be interesting to conduct a reinvestigation over this function to improve the results, for example choosing other plans to convert angles since the plans used in this project have been taken considering the base model of the original NAO robot.

Since it has been defined in development process of the animations, it would be interesting to include more animations or dances that provide greater interaction to therapy and to do more entertaining breaks.

Regarding the interface, it would be possible to add more figures to change the interface model, including a video when the robot performs a dance like the Macarena or including tags or *tooltips* on buttons to improve the feedback to the user.

Bibliografía

[Alcázar et al. 2010] Alcázar, V., Guzmán, C., Prior, D., Borrajo, D., Castillo, L., and Onaindia, E. (2010). Pelea: Planning, learning and execution architecture. In *28th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2010)*. (Citada en pág. 21)

[Alfaro Ballesteros 2012] Alfaro Ballesteros, S. (2012). *Sistema de teleoperación mediante una interfaz natural de usuario*. Bachelor Thesis, Universidad Carlos III de Madrid. (Citada en pág. 38, 43 y 46)

[Burton 2013] Burton, A. (2013). Dolphins, dogs, and robot seals for the treatment of neurological disease. *The Lancet Neurology*, 12(9):851–852. (Citada en pág. 15)

[Calderita et al. 2013] Calderita, L., Bustos, P., Suárez Mejías, C., Fernández, F., and Bandera, A. (2013). Therapist: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*, pages 374–377. (Citada en pág. 6)

[Capek 1921] Capek, K. (1921). *R. U. R. (Rossum's Universal Robots*. Players Press (July 1, 2002). (Citada en pág. 13)

[Castelli 2011] Castelli, E. (2011). Robotic movement therapy in cerebral palsy. *Dev Med Child Neurol*. (Citada en pág. 3)

[Dung et al. 2014] Dung, N. A. and Shimada, A. (2014). A path-planning algorithm for humanoid climbing robot using kinect sensor. In *SICE Annual Conference (SICE), 2014 Proceedings of the*, pages 1549–1554. (Citada en pág. 27)

[Fridin et al. 2011] Fridin, M., Bar-Haim, S., and Belokopytov, M. (2011). Robotics agent coacher for cp motor function (rac cp fun). In *Robotics for Neurology and Rehabilitation, IROS International Conference on Intelligent Robots and Systems*. (Citada en pág. 16)

[Ghallab et al. 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice*. Elsevier. (Citada en pág. 21)

[González et al. 2015] González, J. C., Pulido, J. C., Fernández, F., and Suárez-Mejías, C. (2015). Planning, Execution and Monitoring of Physical Rehabilitation Therapies with a Robotic Architecture. In Cornet, R., Stoicu-Tivadar, L., Hörbst, A., Parra Calderón, C. L., Kjær Andersen, S., and Hercigonja-Szekeres, M., editors, *Proceedings of the 26th Medical Informatics Europe conference (MIE)*, volume 210 of *Studies in Health Technology and Informatics*, pages 339–343, Madrid (Spain). European Federation for Medical Informatics (EFMI), IOS Press. (Citada en pág. 1, 67 y 157)

[Hensey 2009] Hensey, O. (2009). Cerebral palsy: Challenges and opportunities. In *Technical report, State Claims Agency*. (Citada en pág. 4)

[Humberto et al. 2013] Humberto, C., Valdivia, G., Ortega, A. B., Antonio, M., and Salazar, O. (2013). Diseño de un Robot Terapéutico para Miembros Inferiores. *Juarez.AcademiaJournals.com*, 5(1):327–332. (Citada en pág. 14)

[Krägeloh-Mann et al. 2009] Krägeloh-Mann, I. and Cans, C. (2009). Cerebral palsy update. *Brain Dev*, 31(7):537–44. (Citada en pág. 3)

[Limthongthang et al. 2013] Limthongthang, R., Bachoura, A., Songcharoen, P., and Osterman, A. L. (2013). Adult brachial plexus injury: evaluation and management. *The Orthopedic clinics of North America*, 44(4):591–603. (Citada en pág. 3)

[Maheu et al. 2011] Maheu, V., Archambault, P. S., Frappier, J., and Routhier, F. (2011). Evaluation of the jaco robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–5. (Citada en pág. 35)

[Manso et al. 2010] Manso, L., Bachiller, P., Bustos, P., Núñez, P., Cintas, R., and Calderita, L. (2010). Robocomp: A tool-based robotics framework. In Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., and von Stryk, O., editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472 of *Lecture Notes in Computer Science*, pages 251–262. Springer Berlin Heidelberg. (Citada en pág. 24)

[Manzano Carrasco 2016] Manzano Carrasco, C. (2016). *Interacción Humano-Robot con el Robot REEM sobre el Framework RoboComp*. Bachelor Thesis, Universidad Carlos III de Madrid, Madrid (Spain). (Citada en pág. 1, 83 y 85)

[Mardiyanto et al. 2015] Mardiyanto, R., Anggoro, J., and Budiman, F. (2015). 2d map creator for robot navigation by utilizing kinect and rotary encoder. In *Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on*, pages 81–84. IEEE. (Citada en pág. 27)

[Martín et al. 2015] Martín, A., González, J. C., Pulido, J. C., García-Olaya, A., Fernández, F., and Suárez, C. (2015). Therapy Monitoring and Patient Evaluation with Social Robots. In Fardoun, H. M., Gamito, P., Penichet, V. M. R., and Alghazzawi, D. M., editors, *Proceedings of the 3rd Workshop on ICTs for improving Patients Rehabilitation Research Techniques (REHAB)*, pages 152–155, Lisbon (Portugal). ACM. http://www.plg.inf.uc3m.es/~josgonza/files/2015-REHAB-therapy-monitoring-and-patient-evaluation-with-social-robots.pdf. (Citada en pág. 24)

[McDermott et al. 1998] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). Pddl-the planning domain definition language. (Citada en pág. 23)

[Meyer-Heim et al. 2013] Meyer-Heim, A. and van Hedel, H. J. A. (2013). Robot-assisted and computer-enhanced therapies for children with cerebral palsy: current state and clinical implementation. *Semin Pediatr Neurol*, 20(2):139–45. (Citada en pág. 6)

[Rossignoli 2015] Rossignoli, A. (2015). Desarollo de terapias de rehabilitación motora teleoperadas con el robot NAO. *Universidad Carlos III Madrid*. (Citada en pág. 38, 41, 103 y 105)

[Ruiz et al. 2013] Ruiz, E., Acu??a, R., Certad, N., Terrones, A., and Cabrera, M. E. (2013). Development of a control platform for the mobile robot Roomba using ROS and a Kinect sensor. In *Proceedings - 2013 IEEE Latin American Robotics Symposium, LARS 2013*, pages 55–60. (Citada en pág. 35)

[Sale et al. 2012] Sale, P., Lombardi, V., and Franceschini, M. (2012). Hand Robotics Rehabilitation : Feasibility and Preliminary Results of a Robotic Treatment in Patients with Hemiparesis. *Stroke research and treatment*, 2012:6. (Citada en pág. 15)

[Sousa et al. 2011] Sousa, P., Oliveira, J. L., Reis, L. P., and Gouyon, F. (2011). Humanized robot dancing: humanoid motion retargeting based in a metrical representation of human dance styles. In *Progress in Artificial Intelligence*, pages 392–406. Springer. (Citada en pág. 37)

[Suárez Mejías et al. 2013] Suárez Mejías, C., Echevarría, C., Nuñez, P., Manso, L., Bustos, P., Leal, S., and Parra, C. (2013). Ursus: A robotic assistant for training of children with motor impairments. In Pons, J. L., Torricelli, D., and Pajaro, M., editors, *Converging Clinical and Engineering Research on Neurorehabilitation*, volume 1 of *Biosystems & Biorobotics*, pages 249–253. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-34546-3_39. (Citada en pág. 17)

[Tapus et al. 2009] Tapus, A., Tapus, C., and Matarić, M. (2009). The role of physical embodiment of a therapist robot for individuals with cognitive impairments. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 103–107. IEEE. (Citada en pág. 16)

[Turp et al. 2015] Turp, M., Pulido, J. C., González, J. C., and Fernández, F. (2015). Playing with Robots: An Interactive Simon Game. In Puerta, J. M., Gámez, J. A., Dorronsoro, B., Barrenechea, E., Troncoso, A., Baruque, B., and Galar, M., editors, *Proceedings of the 16th Conference of the Spanish Association for Artificial Intelligence (CAEPIA), RSIM workshop*, pages 1085–1095, Albacete (Spain). (Citada en pág. 5)